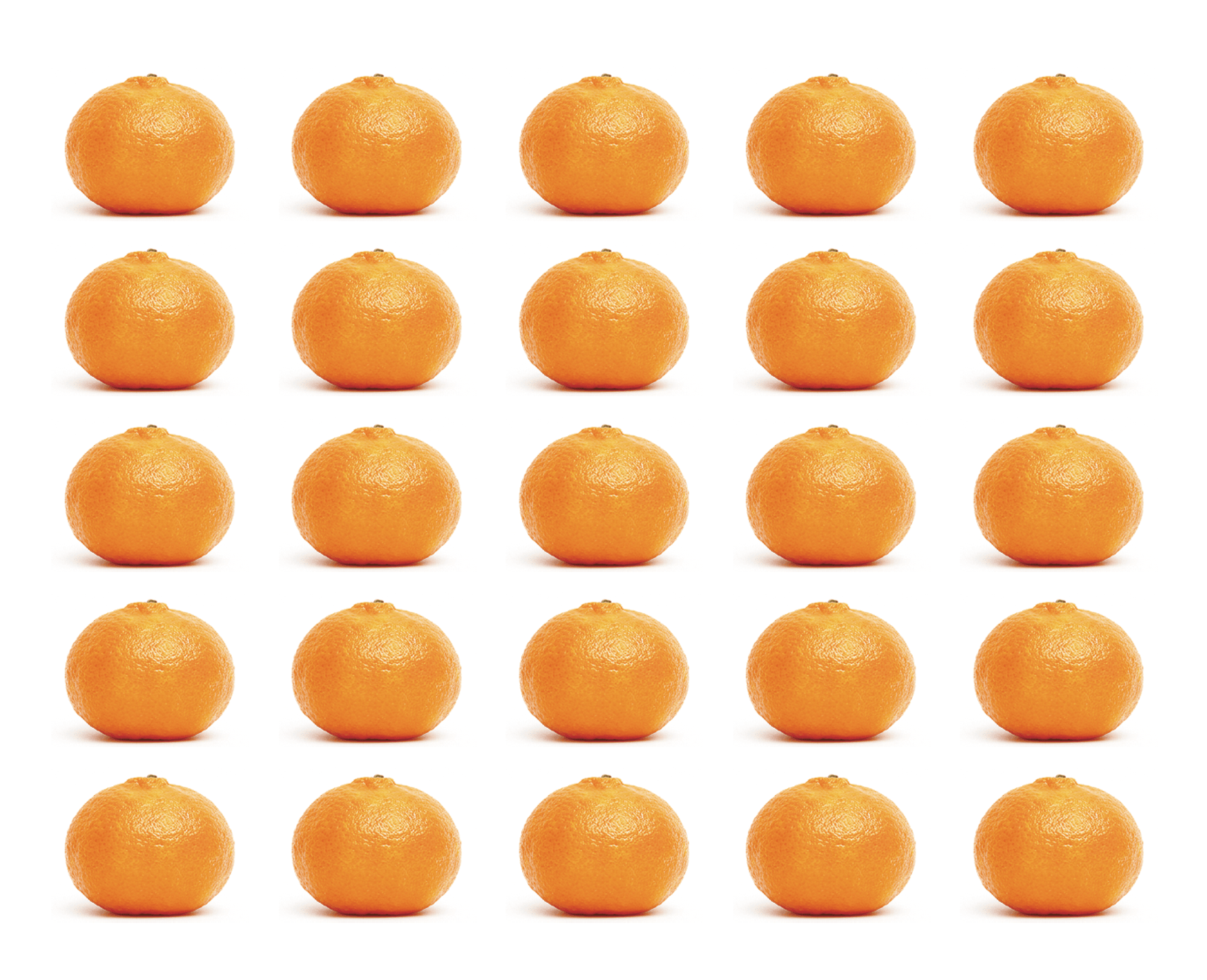
熊平 著

数据挖掘算法与Clementine实践



清华大学出版社

数据挖掘算法与Clementine实践

熊平 著

清华大学出版社
北京

内 容 简 介

本书主要介绍了几种最成熟的数据挖掘方法，并针对每种方法，介绍了应用最广泛的几种实现算法。书中以 Clementine 12.0 为平台，用实例介绍了每种算法的具体应用。全书各章分别介绍了数据挖掘和 Clementine 软件、决策树分类方法（包括 ID3、C4.5、C5.0、CART 等算法）、聚类分析方法（包括 K-Means 算法和 TwoStep 算法）、关联规则挖掘方法（包括 Apriori 算法、CARMA 算法和序列模式挖掘算法）、数据筛选算法（包括特征选择算法和异常检测算法）、回归分析方法（包括线性回归算法和二项 Logistic 回归）、神经网络构建方法（包括多层感知器网络、RBF 网络以及 Kohonen 网络的构建算法）、时间序列分析方法（包括指数平滑法和 ARIMA 模型构建方法）。

图书在版编目（CIP）数据

数据挖掘算法与 Clementine 实践 / 熊平著. —北京：清华大学出版社，2011.4（2017.2 重印）
ISBN 978-7-302-23501-9

Ⅰ. ①数…　Ⅱ. ①熊…　Ⅲ. ①数据采集　Ⅳ. ①TP274

中国版本图书馆 CIP 数据核字（2010）第 157167 号

责任编辑：魏江江　薛　阳
责任校对：时翠兰
责任印制：李红英

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn, http://www.wqbook.com
　　地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
　　社 总 机：010-62770175　　**邮　　购：**010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：虎彩印艺股份有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张：**15.25　　**字　　数：**362 千字
版　　次：2011 年 4 月第 1 版　　**印　　次：**2017 年 2 月第 4 次印刷
印　　数：4801～5100
定　　价：29.00 元

产品编号：038916-02

前言

FOREWORD

21 世纪是信息爆炸的时代。从纷杂无章的数据中发现潜在的知识，进而提供决策支持，是许多企业、部门增强自身竞争力的有力手段。数据挖掘作为重要的知识发现技术，经过几十年的发展，在理论上积累了丰硕的成果，许多高效的、智能的数据挖掘算法被提出并不断得到改进和完善。同时，专用的或者通用的数据挖掘工具也不断被推出和升级。因此，数据挖掘技术在电信、金融、信息安全等许多领域得到了广泛的应用。

本书在内容安排上以理论联系实际为指导原则。在理论上，着重介绍几种最成熟的数据挖掘方法，针对每种方法，则介绍应用最广泛的几种实现算法。在实践上，以 Clementine 12.0 为平台，用实例介绍每种算法的具体应用方法。

本书共分为 9 章。第 1 章和第 2 章分别对数据挖掘和 Clementine 软件进行简要的介绍。第 3～9 章中每章介绍一种数据挖掘方法。第 3 章介绍决策树分类方法，以及构建决策树的 C4.5 算法和 CART 算法；第 4 章介绍聚类分析方法，以及实现聚类分析的 K-Means 算法和 TwoStep 算法；第 5 章介绍关联规则挖掘方法，包括经典的 Apriori 算法、CARMA 算法以及序列模式挖掘算法；第 6 章介绍了两种数据筛选算法，即特征选择算法和异常检测算法；第 7 章介绍了回归分析方法，包括线性回归方法和 Logistic 回归方法；第 8 章介绍了神经网络建模方法，以及用于构建神经网络的多层感知器方法、径向基函数网络构建方法和用于聚类分析的 Kohonen 网络构建方法；第 9 章介绍时间序列分析方法，包括指数平滑法和 ARIMA 模型的构建方法。

本书各章中的实验数据集可在 http://jsjsyzx.znufe.edu.cn/downloads/dataset.rar 下载。

本书得到中央高校基本科研业务费专项资金资助。

由于作者自身水平有限，本书定有不妥及不足之处，恳请读者及专家批评指正。

目录

CONTENTS

第 1 章　数据挖掘概述

1.1 数据挖掘简介

数据挖掘是适应信息社会从海量数据中提取信息的需要而产生的新学科。它是统计学、机器学习、数据库、模式识别、人工智能等学科的交叉，在各行各业的决策支持活动中扮演着越来越重要的角色。

1.1.1　数据、信息和知识

人们在长期的社会生活和实践过程中总结了许多的经验和教训，这些内容对于人们更深层次地了解事物的发展规律有着重要的指导意义。因此人们将生活和实践的经历以及在这些经历中产生的经验和教训以各种形式记录下来，如以文本的形式、音频的形式、视频的形式等。在广义上，把这些记录下来的内容称为数据。

数据仅仅是人们观察客观世界所得到的原始材料，只能够为人们提供关于客观世界最直观的、浅层次的信息。这些浅层次的信息对于深入了解世界和改造世界所起到的作用是非常有限的，它不能构成人们决策或行动的可靠依据。

通过对数据进行分析，找出数据之间所隐含的关系，赋予数据以某种意义和关联，就形成了更深层次的信息。对这些信息进行再加工，即进行更深入的归纳分析，可获得更有用的、能够作为判断、决策和行动依据的信息，这就是知识。人们总是依靠知识来进一步有效地认识世界和改造世界。图 1.1 显示了人们从客观世界中获取知识、又依靠知识继续认识世界的循环过程。

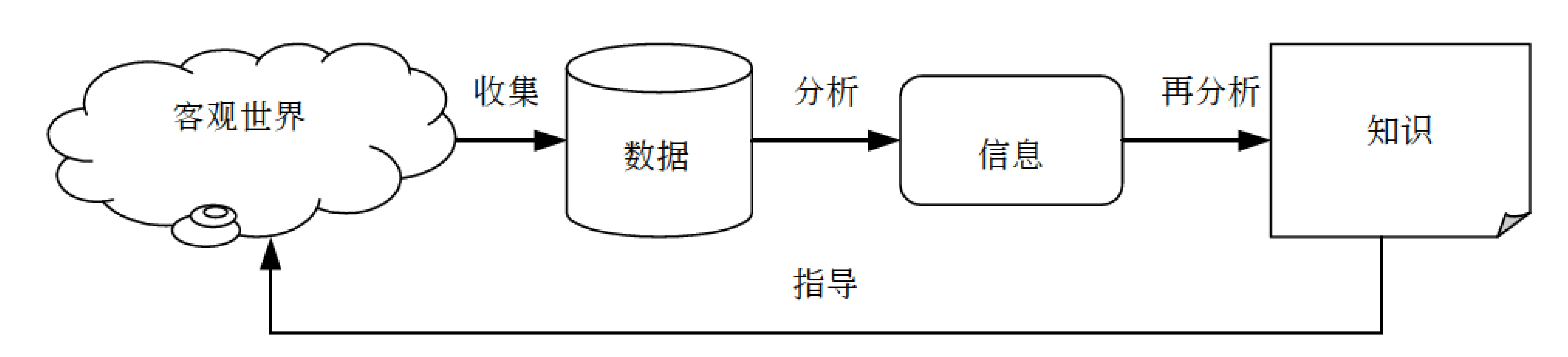


图 1.1　数据、信息与知识

随着人类社会活动越来越频繁和多样化，人们积累的数据和信息也极度增长。人类近 30 年所掌握的信息量占有史以来积累总量的 90%。加之计算机网络的发展，大大促进了数据和信息的流通，使得 21 世纪成为了信息爆炸的时代。

面对海量的数据，人们往往束手无策，逐渐陷入到“丰富的数据”和“贫乏的知识”这个尴尬的境地中。如何从大量的数据中提取知识，是人们目前面临的一个挑战。

为了充分地利用数据，首先必须对数据进行有效的管理，这样就产生了数据库技术。数据库技术可以使人们有条理地记录数据，也能够对数据进行初步的分析和加工。随着数据量的增长，多数据源带来了各种数据格式不相容的问题。为了便于获得决策所需的信息，就有必要将不同格式的数据以统一的形式集中存储在一起，这就形成了数据仓库。

同时，针对数据的复杂化和海量化，在数据分析上也需要效率更高、理论更完善的方法和工具。多年来，数理统计方法以及人工智能和知识工程等领域的研究成果，诸如推理、机器学习、知识获取、模糊理论、神经网络、遗传算法、模式识别、粗糙集理论、支持向量机理论等诸多研究分支，为开发对数据进行深度分析的工具提供了坚实而丰富的理论和技术基础。数据挖掘理论与技术应运而生。

1.1.2 数据挖掘的定义

数据挖掘（Data Mining），又称为数据库知识发现（Knowledge Discovery from Database，KDD），它是一个从大量数据中提取、挖掘出未知的、有价值的模式或规律等知识的复杂过程。它通常与计算机科学有关，并通过统计、在线分析处理、情报检索、机器学习、专家系统（依靠过去的经验法则）和模式识别等诸多方法来实现上述目标。

简而言之，数据挖掘其实是一类深层次的数据分析方法。数据分析本身已经有很多年的历史，但由于以前计算能力的限制，对大数据量进行分析的复杂数据分析方法受到很大限制。现在，由于各行业业务自动化的实现，产生了大量的业务数据。同时，由于计算机性能的不断提高，各种智能分析理论的日趋成熟，使得对海量数据进行分析，并为决策提供真正有价值的信息成为一个有效提高企业、组织竞争力的有效途径。

因此，数据挖掘可以描述为：按既定决策目标，对大量的数据进行探索和分析，揭示隐藏的、未知的或验证已知的规律性，并进一步将其模型化的先进有效的方法。

数据挖掘与传统的数据分析（如查询、报表、联机应用分析）又有本质区别。数据挖掘是在没有明确假设的前提下去挖掘信息、发现知识。同时，数据挖掘所得到的信息应具有先前未知、有效和可实用 3 个特征。先前未知的信息是指该信息是预先未曾预料到的，即数据挖掘是要发现那些不能靠直觉发现的信息或知识，甚至是违背直觉的信息或知识，挖掘出的信息越是出乎意料，反而可能越有价值。

1.2 数据挖掘过程

数据挖掘是通过自动或半自动化的工具对大量数据进行探索和分析的过程，其目的是发现其中有意义的模式和规律。就数据挖掘的流程来看，一个数据挖掘过程究竟应该包含哪些基本的步骤，并没有一个统一的、通用的过程模型。这方面的过程模型较多，比较权威的有 SPSS 的 5A 法，即访问（Access）、分析（Analyze）、评估（Assess）、行动（Action）、自动化（Automate）；SAS 的 SEMMA 法，即抽样（Sample）、探索（Explore）、建模（Model）、修正（Modify）、评估（Assess）。

目前，数据挖掘领域最权威的过程模型是 CRISP-DM 模型，它是目前事实上最权威的行业标准。CRISP-DM（Cross-Industry Standard Process for Data Mining），即“跨行业

数据挖掘过程标准”。

在 1996 年时，数据挖掘市场是年轻而不成熟的，但是这个市场显示了爆炸式的增长。3 个在这方面经验丰富的公司 DaimlerChrysler、SPSS、NCR 发起建立一个社团，目的是建立数据挖掘方法和过程的标准。在获得了 EC（European Commission）的资助后，他们开始实现他们的目标。为了征集业界广泛的意见共享知识，他们创建了 CRISP-DM Special Interest Group（SIG）。

1999 年，SIG 组织开发并提炼出 CRISP-DM，同时在 Mercedes-Benz 和 OHRA（保险领域）企业进行了大规模数据挖掘项目的实际试用。SIG 还将 CRISP-DM 和商业数据挖掘工具集成起来。2000 年，CRISP-DM 1.0 版正式推出，应该说 CRISP-DM 是实际项目的经验总结和理论抽象。CRISP-DM 强调，数据不单是数据的组织或者呈现，也不仅是数据分析和统计建模，而是一个从理解业务需求、寻求解决方案到接受实践检验的完整过程。通过不断的发展，CRISP-DM 模型在各种 KDD 过程模型中占据领先位置，采用率达到近 60%（Cios and Kurgan，*Trends in data mining and knowledge discovery*，2005）。

CRISP-DM 模型定义了 6 个阶段，分别是：商业理解（Business Understanding）、数据理解（Data Understanding）、数据准备（Data Preparation）、建立模型（Modeling）、模型评估（Evaluation）、结果部署（Deployment）。

CRISP-DM 模型如图 1.2 所示。

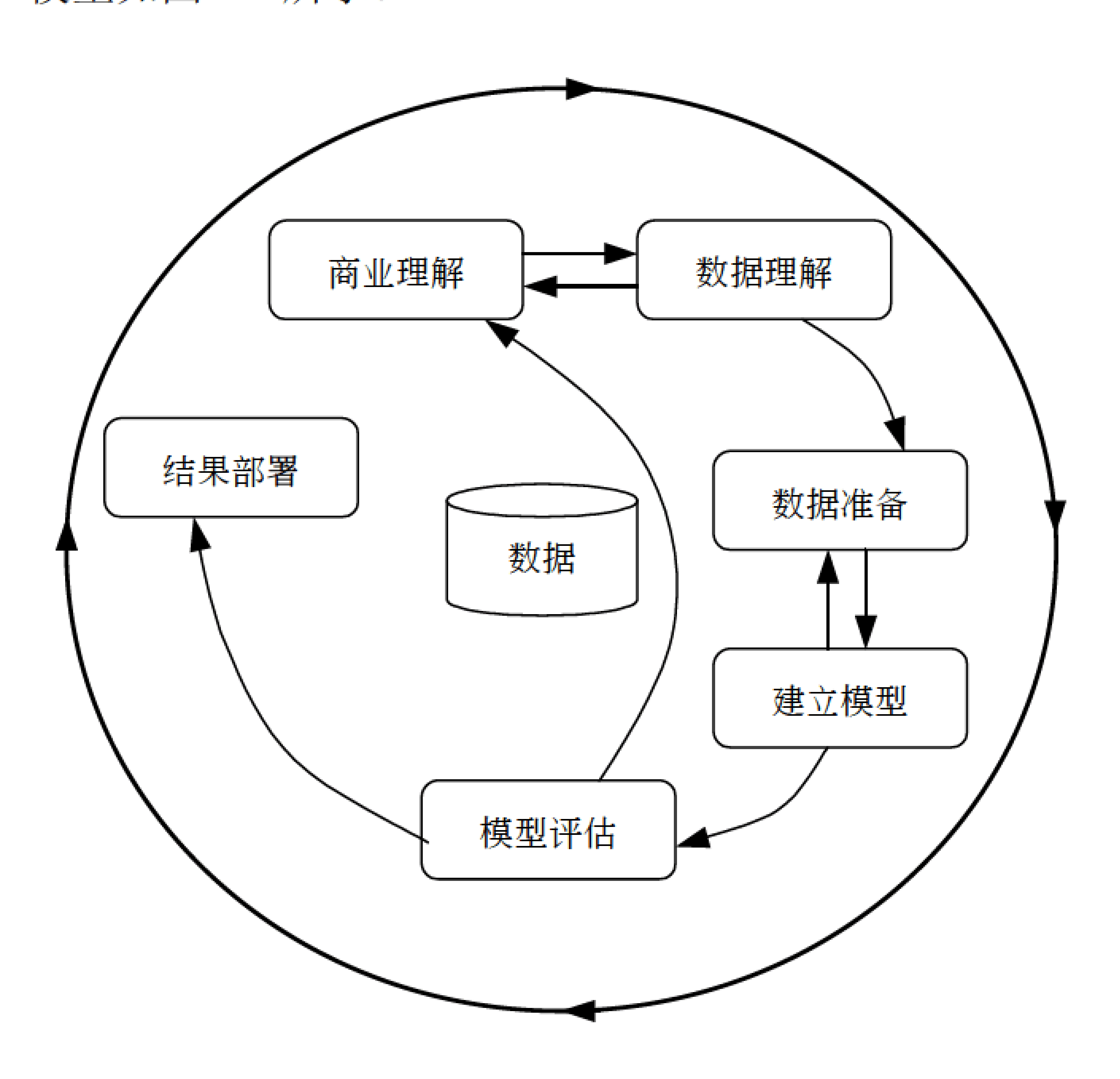


图 1.2　CRISP-DM 模型

CRISP-DM 模型为一个 KDD 工程提供了一个完整的过程描述。一个数据挖掘项目的生命周期包含 6 个阶段。这 6 个阶段的顺序是不固定的，经常需要前后调整这些阶段，这依赖每个阶段或是阶段中特定任务的产出物是否是下一个阶段必需的输入。图 1.2 中

箭头指出了最重要的和依赖度高的阶段关系。

图 1.2 的外圈象征数据挖掘自身的循环本质——在一个解决方案发布之后一个数据挖掘的过程才可以继续。在这个过程中得到的知识可以触发新的过程，经常是更聚焦的商业问题。后续的过程可以从前一个过程得到益处。

1．商业理解（Business Understanding）

最初的阶段集中在理解项目目标和从业务的角度理解需求，同时将这个知识转化为数据挖掘问题的定义和完成目标的初步计划。

2．数据理解（Data Understanding）

数据理解阶段从初始的数据收集开始，通过一些活动的处理，目的是熟悉数据，识别数据的质量问题，首次发现数据的内部属性，或是探测引起兴趣的子集去形成隐含信息的假设。

3．数据准备（Data Preparation）

数据准备阶段包括从未处理的数据中构造最终数据集的所有活动。这些数据将是模型工具的输入值。这个阶段的任务有的需要执行多次，没有任何规定的顺序。任务包括表、记录和属性的选择，以及为模型工具转换和清洗数据。

4．建立模型（Modeling）

在这个阶段，可以选择和应用不同的模型技术，模型参数被调整到最佳的数值。一般，有些技术可以解决一类相同的数据挖掘问题。有些技术在数据形成上有特殊要求，因此需要经常跳回到数据准备阶段。

5．模型评估（Evaluation）

到项目的这个阶段，已经从数据分析的角度建立了一个高质量的模型。在开始最后部署模型之前，需要彻底地评估模型，检查构造模型的步骤，确保模型可以完成业务目标。这个阶段的关键目的是确定是否有重要业务问题没有被充分的考虑。在这个阶段结束后，将决定一个数据挖掘结果是否可以付诸使用。

6．结果部署（Deployment）

通常，模型的创建不是项目的结束。模型的作用是从数据中找到知识，获得的知识需要以便于用户使用的方式重新组织和展现。根据需求，这个阶段可以产生简单的报告，或是实现一个比较复杂的、可重复的数据挖掘过程。在很多案例中，这个阶段是由客户而不是数据分析人员承担部署的工作。

需要注意的是，以上 6 个步骤并非完全按照此顺序来执行。在应用中，应该针对不同的应用环境和实际情况做出必要的调整。例如，数据准备通常在建模之前进行。但在建模阶段所作的决策以及所收集的信息通常又会给人们一些反馈，并引导人们重新考虑数据准备阶段的部分，如此一来可能会出现新的建模问题。这两个阶段互相反馈，直到

两个阶段都得以充分解决。与之相似，评估过程可能会引导人们重新评估最初的商业理解，这时，可能会修正此前的商业理解，并制订一个更好的目标，然后重新进行该过程的其余部分。

另外，一个数据挖掘项目通常并不是一次性地执行了上述 6 个步骤就结束了，它往往是一个反复迭代、不断完善的过程。从一个数据挖掘循环获得的知识几乎无所例外地会导致新的问题、新的争论以及新的机会来识别和满足客户的需求。这些新问题、新争论和新机会通常可以通过再次挖掘数据得以解决。这个挖掘和识别新机会的过程不仅应该成为人们考虑业务的方式的组成部分，还应该成为整个业务策略的基石。

1.3 数据挖掘方法

数据挖掘所涉及的学科领域和方法很多，目前比较成熟且应用广泛的方法主要有分类方法、聚类分析、关联规则分析、异常检测、预测方法等。

1. 分类

分类（Classification）是一种数据分析形式，它可以用来抽取能够描述重要数据集合的模型。分类方法用于预测数据对象的离散类别（Categorical Label）。例如，根据一个客户的“年龄”、“性别”、“职业”、“收入情况”等属性的取值，来判定该客户“信用等级”的取值是“优”、“良”或者“差”，这就是一个典型的分类问题。

分类技术在很多领域都有应用，例如可以通过客户分类构造一个分类模型来对银行贷款进行风险评估；当前的市场营销中很重要的一个特点是强调客户细分。客户类别分析的功能也在于此，采用数据挖掘中的分类技术，可以将客户分成不同的类别，比如呼叫中心在设计时可以分为：呼叫频繁的客户、偶然大量呼叫的客户、稳定呼叫的客户、其他。这样就可以帮助呼叫中心寻找出这些不同种类客户之间的特征。这样的分类模型可以让用户了解不同行为类别客户的分布特征；其他分类应用如文献检索和搜索引擎中的自动文本分类技术，安全领域有基于分类技术的入侵检测等。机器学习、专家系统、统计学和神经网络等领域的研究人员已经提出了许多具体的分类方法。

本书第 3 章介绍了决策树分类方法，以及用来实现决策树方法的几种算法，包括 ID3、C4.5、C5.0、CART 算法。

2. 聚类分析

聚类分析是根据“物以类聚”的道理，对样本进行分类的一种多元统计分析方法，其处理的对象是大量的样本，要求能合理地按各自的特性来进行合理的分类，没有任何模式可供参考和指导，即是在没有先验知识的情况下进行的。聚类分析起源于分类学，在古老的分类学中，人们主要依靠经验和专业知识来实现分类，很少利用数学工具进行定量的分类。随着人类科学技术的发展，对分类的要求越来越高，以致有时仅凭经验和专业知识难以确切地进行分类，于是人们逐渐地把数学工具引用到了分类学中，形成了数值分类学，之后又将多元分析的技术引入到数值分类学形成了聚类分析。

聚类是将数据划分到不同的类或者簇这样的一个过程，所以同一个簇中的对象有很

大的相似性，而不同簇间的对象有很大的相异性。聚类分析之后所得到的类或者簇在事先是不知道的，这也是聚类和分类之间的主要区别，即分类是有先验知识指导的，而聚类没有先验知识的指导。

聚类分析被应用于很多方面，在商业上，聚类分析被用来发现不同的客户群，并且通过购买模式刻画不同的客户群的特征；在生物上，聚类分析被用于动植物分类和对基因进行分类，获取对种群固有结构的认识；在因特网应用上，聚类分析被用来在网上进行文档归类来修复信息。

本书第 4 章介绍了聚类分析方法，以及用来实现聚类的分析方法 K-Means 和 TwoStep 算法。另外，在第 8 章中介绍的 Kohonen 网络也可以用于聚类分析。

3．关联规则分析

关联规则挖掘是由 Rakesh Apwal 等人首先提出的。两个或两个以上变量的取值之间存在某种规律性，就称为关联。数据关联是数据库中存在的一类重要的、可被发现的知识。关联分为简单关联、时序关联和因果关联。把寻找到的关联用指定的规则方式表达出来，就形成了关联规则，一般用支持度和置信度两个阈值来度量关联规则的相关性，还不断引入兴趣度、相关性等参数，使得所挖掘的规则更符合需求。

例如关联规则：牛奶∧果冻⇒花生酱 [support=22%，confidence=50%]，表示有 22%的顾客同时购买了牛奶和果冻，这些顾客中有 50%还同时购买了花生酱。

通俗地说，对于一个关系型数据表而言，关联规则分析就是要找到潜藏在数据集中属性（字段）之间的关系。关联规则分析是一种无监督学习，即没有先验知识指导下的分析方法。

“啤酒与尿布”是我们耳熟能详的故事，这就是一个经典的关联规则分析过程。通过对超市购物数据集的挖掘，发现年轻的父亲在给自己的婴儿购买尿布时会捎带着为自己购买几罐啤酒，这一规则使得超市可以将尿布和啤酒的货架放在一起，或者将两者捆绑销售。正因为如此，关联规则分析有时也被称为“购物篮分析”。显然，关联规则分析并不仅仅限于购物篮分析。

序列模式可以看做一种特殊的关联规则。序列模式挖掘的目的是要找出类似于“5%的客户先买床，再买床垫，最后买枕头”这样的规律。显然，序列模式挖掘的侧重点是事件发生的先后次序。序列模式在 Web 使用挖掘（Web Usage Mining）中也是非常有用的，比如用来分析在服务器日志中的点击流（Clickstreams）。

本书第 5 章介绍了关联规则分析方法，以及用来实现关联规则分析的 Apriori 算法、CARMA 算法以及序列模式挖掘算法。

4．异常检测

任何事物的发展与变化都有其必然性和偶然性。有些情况下，我们的研究重点并非要挖掘出具有普遍性和通用性的规律，而在于隐藏在数据中的异常信息（outlier）。例如在某个季节里，某一天的气温很高或很低，这个温度数据就是一个异常。有时候异常信息更能反映出事物的本质规律。

异常是在数据集中与众不同的数据，使人怀疑这些数据并非随机偏差，而是产生于

完全不同的机制。它的行为与正常的行为有显著的不同。异常检测和分析是数据挖掘中一个重要的方面，也是一个非常有意义的挖掘课题。它用来发现“小的模式”，即数据集中显著不同于其他数据的对象。异常检测具有广泛的应用，如电信和信用卡欺骗、贷款审批、药物研究、医疗分析、消费者行为分析、气象预报、金融领域客户分类、网络入侵检测等。

异常检测的方法大致可以分为 4 类：基于统计（statistical-based）的方法、基于距离（distance-based）的方法、基于偏差（deviation-based）的方法、基于密度（density-based）的方法。

本书第 6 章介绍了异常检测的相关内容和实现算法。

5．预测

预测就是根据历史数据和当前数据来推测出未来数据的一种挖掘方法。和分类一样，预测也是一种有指导的学习过程。不同的是，分类用于处理离散型数据，而预测用于分析连续型数据。从这个意义上来讲，很多数据挖掘方法都可以划归到预测方法中，例如回归分析、神经网络、时间序列分析等。

回归分析（Regression Analysis）是确定两个或两个以上变量间相互依赖的定量关系的一种统计分析方法，应用十分广泛。按照涉及的自变量的多少，回归分析可分为一元回归分析和多元回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。本书第 7 章介绍了最常用的线性回归与二项 Logistic 回归分析方法。

神经网络是依据人类大脑信息处理的特点，进行分布式并行信息处理的数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。神经网络具有自学习功能、联想存储功能和高速寻找优化解的能力，在经济预测、市场预测、效益预测等领域具有良好的预测效果。本书第 8 章介绍了多层感知器与 RBF 网络两种预测型的神经网络。

时间序列分析是根据系统观测得到的时间序列数据，通过曲线拟合和参数估计来建立数学模型的理论和方法。其基本原理是承认事物发展的延续性，应用过去的数据，就能推测事物的发展趋势，同时考虑到事物发展的随机性。时间序列分析常用在国民经济宏观控制、区域综合发展规划、企业经营管理、市场潜量预测、气象预报、水文预报、地震前兆预报、农作物病虫灾害预报、环境污染控制、生态平衡、天文学和海洋学等方面。本书第 9 章介绍了指数平滑法、ARIMA 模型等时间序列预测方法。

1.4 数据挖掘工具及软件

古人云：“工欲善其事，必先利其器”，说的是要想把工作完成，做得完善，应该先把工具准备好。数据挖掘所处理的往往是海量的数据集，没有高效的挖掘工具和软件是难以实现的。

数据挖掘工具根据其适用的范围可分为专用挖掘工具和通用挖掘工具。

专用数据挖掘工具是针对某个特定领域的问题提供解决方案，在涉及算法的时候充分考虑了数据、需求的特殊性，并作了优化。对任何领域，都可以开发特定的数据挖掘

工具。例如，IBM 公司的 AdvancedScout 系统针对美国职业篮球联赛（NBA）的数据，帮助教练优化战术组合。特定领域的数据挖掘工具针对性比较强，只能用于一种应用。也正因为针对性强，往往采用特殊的算法，可以处理特殊的数据，实现特殊的目的，发现的知识可靠度也比较高。

通用数据挖掘工具不区分具体数据的含义，采用通用的挖掘算法，处理常见的数据类型。例如，IBM 公司 Almaden 研究中心开发的 QUEST 系统，SGI 公司开发的 MineSet 系统，加拿大 SimonFraser 大学开发的 DBMiner 系统，SPSS 公司的 Clementine 等。通用的数据挖掘工具可以做多种模式的挖掘，挖掘什么、用什么来挖掘都由用户根据自己的应用来选择。

下面介绍几款优秀的通用数据挖掘工具软件。

1. QUEST

QUEST 是 IBM 公司 Almaden 研究中心开发的一个多任务数据挖掘系统，目的是为新一代决策支持系统的应用开发提供高效的数据挖掘基本构件。该系统具有如下特点：

① 提供了专门在大型数据库上进行各种挖掘的功能：关联规则发现、序列模式发现、时间序列、聚类、决策树分类、递增式主动挖掘等。

② 各种挖掘算法具有近似线性计算复杂度，可适用于任意大小的数据库。算法具有找全性，即能将所有满足指定类型的模式全部寻找出来。

③ 为各种发现功能设计了相应的并行算法。

2. MineSet

MineSet 是由 SGI 公司和美国 Standford 大学联合开发的多任务数据挖掘系统。MineSet 集成多种数据挖掘算法和可视化工具，帮助用户直观地、实时地发掘、理解大量数据背后的知识。MineSet 有如下特点：

① MineSet 以先进的可视化显示方法闻名于世。

② 支持多种关系数据库。可以直接从 Oracle、Informix、Sybase 的表读取数据，也可以通过 SQL 命令执行查询。

③ 多种数据转换功能。在进行挖掘前，MineSet 可以去除不必要的数据项，统计、集合、分组数据，转换数据类型，构造表达式由已有数据项生成新的数据项，对数据采样等。

④ 操作简单、支持国际字符、可以直接发布到 Web。

3. DBMiner

DBMiner 是加拿大 SimonFraser 大学开发的一个多任务数据挖掘系统，它的前身是 DBLearn。该系统设计的目的是把关系数据库和数据开采挖掘集成在一起，以面向属性的多级概念为基础发现各种知识。DBMiner 系统具有如下特色：

① 能完成多种知识的发现：泛化规则、特性规则、关联规则、分类规则、演化知识、异常知识等。

② 综合了多种数据挖掘技术：面向属性的归纳、统计分析、逐级深化发现多级规则、

元规则引导发现等方法。

③ 提出了一种交互式的类 SQL 语言——数据挖掘查询语言 DMQL。

④ 能与关系数据库平滑集成。

⑤ 实现了基于客户-服务器体系结构的 UNIX 和 PC（Windows/NT）版本的系统。

4. Intelligent Miner

由美国 IBM 公司开发的数据挖掘软件 Intelligent Miner 是一种分别面向数据库和文本信息进行数据挖掘的软件系列，它包括 Intelligent Miner for Data 和 Intelligent Miner for Text。Intelligent Miner for Data 可以挖掘包含在数据库、数据仓库和数据中心中的隐含信息，帮助用户利用传统数据库或普通文件中的结构化数据进行数据挖掘。它已经成功应用于市场分析、诈骗行为监测及客户关系管理等；Intelligent Miner for Text 允许企业通过文本信息进行数据挖掘，文本数据源可以是文本文件、Web 页面、电子邮件、Lotus Notes 数据库等。

5. SAS Enterprise Miner

这是一种在我国的企业中得到采用的数据挖掘工具，比较典型的包括上海宝钢配矿系统应用和铁路部门在春运客运研究中的应用。SAS Enterprise Miner 是一种通用的数据挖掘工具，按照“抽样-探索-转换-建模-评估”的方法进行数据挖掘。可以与 SAS 数据仓库和 OLAP 集成，实现从提出数据、抓住数据到得到解答的“端到端”知识发现。

6. SPSS Clementine

SPSS Clementine 是一个开放式数据挖掘工具，曾两次获得英国政府 SMART 创新奖，它不但支持整个数据挖掘流程，从数据获取、转化、建模、评估到最终部署的全部过程，还支持数据挖掘的行业标准——CRISP-DM。Clementine 的可视化数据挖掘使得“思路”分析成为可能，即将精力集中在要解决的问题本身，而不是局限于完成一些技术性工作（比如编写代码）。提供了多种图形化技术，有助理解数据间的关键性联系，指导用户以最便捷的途径找到问题的最终解决办法。

本书采用 Clementine 12.0 为数据挖掘工具，重点介绍了 Clementine 的用法。在每章介绍了相关的数据挖掘算法之后，通过实例阐述了利用 Clementine 进行数据挖掘的详细过程。

第 2 章　Clementine 概述

Clementine 是 ISL（Integral Solutions Limited）公司开发的数据挖掘工具平台。1999 年 SPSS 公司收购了 ISL 公司，对 Clementine 产品进行重新整合和开发，现在 Clementine 已经成为 SPSS 公司的又一亮点。目前，Clementine 已经发展到 13.0 版本，SPSS 公司将其更名为 PASW Modeler，但由于“Clementine”一词在数据挖掘领域已经深入人心，故本书仍然沿用这一名称。

作为一个数据挖掘平台，Clementine 结合商业技术可以快速建立预测性模型，进而应用到商业活动中，帮助人们改进决策过程。强大的数据挖掘功能和显著的投资回报率使得 Clementine 在业界久负盛誉。Clementine 中功能强大的数据挖掘算法，使数据挖掘贯穿业务流程的始终，在缩短投资回报周期的同时极大地提高了投资回报率。

2.1　Clementine 简介

Clementine 使用客户端/服务器体系结构将资源集约型操作的请求分发给功能强大的服务器软件，因而使大数据集的传输速度大大加快。

Clementine Client 是具有完整功能的产品，它安装并运行于用户的计算机上。它既可以在本机模式下独立运行，也可以与 Clementine Server 一起联机使用，从而提高了对大数据集的处理速度。通常，数据挖掘人员可以通过客户端软件来完成所有的工作。因此，本书仅介绍 Clementine 客户端，且后续章节中的所有示例均在 Clementine 12.0 中执行。

Clementine 自带有丰富的说明文档。安装 Clementine 12.0 之后，可以通过 Windows“开始”→SPSS Inc→Clementine 12.0→Documentation 级联菜单进入存放文档的目录。这些文档是应用 Clementine 时的重要参考资料，包括：

ClementineUsersGuide.pdf：Clementine 的一般使用介绍，包括如何构建数据流、处理缺失值、生成 CLEM 表达式、处理项目和报告以及将用于部署的流打包为 SPSS Predictive Enterprise Services 或预测应用程序。

ClementineSourceProcessOutputNodes.pdf：介绍用于以不同的格式读取、处理和输出数据的所有节点。

ClementineModelingNodes.pdf：Clementine 提供了各种借助机器学习、人工智能和统计学的建模方法。

ClementineApplicationsGuide.pdf：该指南中的示例旨在为具体的建模方法和技术提供具有针对性的简介。用户也可以在“帮助”菜单中查阅该指南的在线版本。

ClementineScriptingAutomation.pdf：通过编写脚本和执行批处理模式实现系统自动化的相关信息，包括用于操作节点和流的属性信息。

ClementineCLEFReference.pdf：CLEF 提供了将第三方程序（例如，数据处理例程

或建模算法）作为节点集成到 Clementine 的功能。

ClementineDatabaseMiningGuide.pdf：有关如何利用数据库的功能通过第三方算法来改进性能并增强分析功能的信息。

ClementineServerandPerformanceGuide.pdf：有关如何配置和管理 Clementine Server 的信息。

ClementineAlgorithmsGuide.pdf：介绍 Clementine 中所用建模方法的数学基础。

ClementineSolutionPublisher.pdf：Clementine Solution Publisher 是一个附加式组件，通过它组织可发布在标准 Clementine 环境之外使用的流。

CRISP-DM.pdf：借助 CRISP-DM 方法进行数据挖掘的分步指南。

SPSSCommandSyntaxReference.pdf：SPSS 命令语法的文档（可通过 Clementine 中的 SPSS 转换和 SPSS 输出节点获取）。

2.2 Clementine 基本操作

2.2.1 Clementine 主窗口

通过 Windows“开始”→SPSS Inc→Clementine 12.0→Clementine 12.0 级联菜单，可以打开 Clementine 12.0 的主窗口，如图 2.1 所示。



图 2.1 Clementine 主窗口

主窗口中主要包括数据流区域、选项板区、管理器窗口、工程管理区。

1. 数据流区域

Clementine 在进行数据挖掘时是基于数据流形式的，从读入数据到最后的结果显示

都是通过流程图的形式显示在数据流区域内。数据流（stream）由一系列的节点构成，每个节点代表了对数据的某种处理，节点之间通过有方向的箭头连接。因此，将各种操作节点组合在一起便形成了一条通向目标的路径。

数据流区域是整个操作界面中最大的部分，整个建模过程以及对模型的操作都将在这个区域内执行。可以通过“文件”→“新建流”菜单新建一个空白的数据流，也可以打开已有的数据流。所有在一个运行期内打开的数据流都将保存在管理器窗口的“流”标签下。

2．选项板区

选项板区横跨于 Clementine 操作界面的下部，它被分为收藏夹、源、记录选项、字段选项、图形、建模、输出、导出 8 个栏目，其中每个栏目包含了具有相关功能的节点。

节点是数据流的基本组成部分，每一个节点拥有不同的数据处理功能。设置不同的栏是为了将不同功能的节点分组，下面介绍各个栏的作用。

收藏夹：该栏放置了用户经常使用的节点，方便用户操作。用户可以自定义其收藏夹栏，操作方法为：选中菜单栏的“工具”→“管理选项板”子菜单，在弹出的“选项板管理器”中选中“收藏夹”，然后单击按钮，即可对收藏夹进行修改。

源：该栏包含了能读入数据到 Clementine 的节点。例如“可变文件”节点读取自由格式的文本文件到 Clementine，“SPSS 文件”读取 SPSS 文件到 Clementine。

记录选项：该栏包含的节点能对数据记录进行操作。例如“选择”节点用来筛选出满足条件的记录，“合并”节点将来自不同数据源的数据合并在一起，“追加”节点可以向数据文件中添加记录等。

字段选项：该栏包含了能对字段进行操作的节点。例如“过滤”节点能让被过滤的字段不作为建模过程的输入，“导出”节点能根据现有的字段来生成新的字段。

图形：该栏包含了众多的图形节点，这些节点用于在建模前或建模后将数据通过图形形式输出，便于用户更直观地了解数据特性。

建模：该栏包含了各种已封装好的模型，例如神经网络、决策树（C5.0）等。这些模型能完成预测、分类、关联分析等功能。

输出：该栏提供了许多能输出数据、模型结果的节点，用户不仅可以直接在 Clementine 中查看输出结果，也可以输出到其他应用程序中查看，例如 SPSS 和 Excel。

导出：该栏下的节点提供了将数据以各种不同的格式导出的功能，作为与其他软件的接口。例如“数据库”节点可以将数据写入 ODBC 数据源。为了将数据写入 ODBC 数据源，该数据源必须存在，同时用户必须具有写权限。

3．管理器窗口

管理器窗口中共包含了“流”、“输出”、“模型”3 个栏。其中“流”中放置了运行期内打开的所有数据流，可以通过右击数据流名对数据流进行保存、设置属性等操作。“输出”中包含了运行数据流时所有的输出结果，可以通过双击结果名查看输出的结果。“模型”中包含了模型的运行结果，可以右击该模型从弹出的“浏览”中查看模型结果，也可以将模型结果加入到数据流中。

4．工程管理区

工程管理区含有两个选项栏，一个是 CRISP-DM，一个是“类”。

CRISP-DM 的设置是基于 CRISP-DM 模型的思想，它方便用户存放在挖掘各个阶段形成的文件。右击阶段名，可以选择生成该阶段要拥有的文件，也可以打开已存在的文件将其放入该阶段。这样做的好处是使用户对数据挖掘过程一目了然，也有利于对它进行修改。

“类”窗口具有同 CRISP-DM 窗口相似的作用，它的分类不是基于挖掘的各个过程，而是基于存储的文件类型，例如数据流文件、节点文件、图表文件等。

2.2.2 数据流的基本操作

1．生成数据流的基本过程

数据流由一系列的节点组成，当数据通过每个节点时，节点对它进行定义好的操作。建立数据流通常遵循以下 4 个步骤：

（1）向数据流区域增添新的节点；

（2）将这些节点连接到数据流中；

（3）设定数据节点或数据流的功能；

（4）运行数据流。

2．向数据流区域添加/删除节点

当向数据流区域添加新的节点时，通过下面 3 种方法均可实现：

（1）双击选项板区中待添加的节点。

（2）单击待添加节点，按住鼠标不放，将其拖入数据流区域内。

（3）先选中选项板区中待添加的节点，然后将鼠标放入数据流区域，在鼠标变为十字形时单击数据流区域的任何空白处。

通过上面 3 种方法都将发现选中的节点出现在了数据流区域内。

当不再需要数据流区域内的某个节点时，可以通过以下两种方法来删除：

（1）单击待删除的节点，按 Delete 键删除。

（2）右击待删除的节点，在快捷菜单中选择 delete 命令。

注意，删除一个节点后，与之相连的所有连接也将一并被删除。

3．将节点连接到数据流中

上面介绍了将节点添加到数据流区域的方法，然而要使节点真正发挥作用，需要把节点连接到数据流中。以下有 3 种方法可将节点连接到数据流中：

（1）双击节点。首先选中数据流中要连接新节点的节点（起始节点），然后双击选项板区中要添加到数据流的节点（目标节点），这样新节点会出现在数据流区域，并自动建立从起始节点到目标节点的连接。

（2）通过 Alt 键连接。首先在数据流中选中连接起始节点，按住 Alt 键不放，然后用鼠标将起始节点拖曳到目标节点后放开，连接便自动生成。

（3）手动连接。右击待连接的起始节点，从弹出的快捷菜单中选择“连接...”命令，这时鼠标会变成形状，然后用鼠标单击目标节点，连接便自动生成。

需要注意的是，并不是任何两个节点之间都可以建立连接。

4．绕过数据流中的节点

当暂时不需要数据流中的某个节点时可以绕过该节点。在绕过它时，如果该节点既有输入节点又有输出节点，那么它的输入节点和输出节点便直接相连；如果该节点没有输出结点，那么绕过该节点时与这个节点相连的所有连接便被取消。

方法：按住 Alt 键不放，然后双击数据流中需要绕过的节点，如图 2.2 所示。



（a）绕过节点之前　　（b）绕过节点之后

图 2.2　绕过节点前后对比

5．将节点插入已存在的连接中

当需要在两个已连接的节点中再插入一个节点时，可以用鼠标将连接拖到要插入的节点上，即可将节点插入到连接中，如图 2.3 所示。同时原来两个节点间的连接被删除。

6．删除连接

当某个连接不再需要时，可以通过以下两种方法将其删除：

（1）选择待删除的连接，右击，从弹出的快捷菜单中选择“删除连接”命令。

（2）选择待删除连接的节点，按 F3 键，删除所有连接到该节点上的连接。

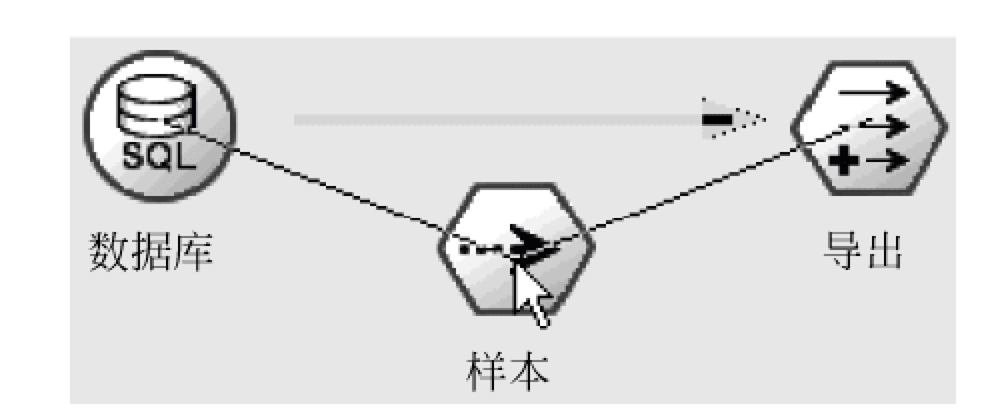


图 2.3　在连接中插入新节点

7．数据流的执行

数据流构建好后要通过执行数据流数据才能从读入开始流向各个数据节点。执行数据流的方法有以下 3 种。

（1）单击菜单栏中的▶按钮，数据流区域内的所有数据流将被执行。

（2）先选择要输出的数据流，再单击菜单栏中的▶按钮，被选的数据流将被执行。

（3）选择要执行的数据流中的输出节点，右击，在弹出的快捷菜单中选择“执行”命令，执行被选中的节点。

第3章　决　策　树

3.1　分类与决策树概述

3.1.1　分类与预测

分类是一种应用非常广泛的数据挖掘技术，应用的例子也非常多。例如，根据信用卡支付历史记录，来判断具备哪些特征的用户往往具有良好的信用；根据某种病症的诊断记录，来分析哪些药物组合可以带来良好的治疗效果。这些过程的一个共同特点是：根据数据的某些属性，来估计一个特定属性的值。例如在信用分析案例中，根据用户的“年龄”、“性别”、“收入水平”、“职业”等属性的值，来估计该用户“信用度”属性的值应该取“好”还是“差”，在这个例子中，所研究的属性“信用度”是一个离散属性，它的取值是一个类别值，这种问题在数据挖掘中被称为**分类**。

还有一种问题，例如根据股市交易的历史数据来估计下一个交易日的大盘指数，这里所研究的属性“大盘指数”是一个连续属性，它的取值是一个实数。那么这种问题在数据挖掘中被称为**预测**。

总之，当估计的属性值是离散值时，这就是分类；当估计的属性值是连续值时，这就是预测。

本章研究的问题是分类。解决分类问题有很多方法，例如基于统计的方法、基于距离的方法、基于决策树的方法和基于神经网络的方法等。其中，决策树是应用最为广泛的方法。

3.1.2　决策树的基本原理

1．构建决策树

通过一个实际的例子，来了解一些与决策树有关的基本概念。

表3-1是一个数据库表，记载着某银行的客户信用记录，属性包括“姓名”、“年龄”、“职业”、“月薪”、……、“信用等级”，每一行是一个客户样本，每一列是一个属性（字段）。这里把这个表记做数据集 D。

银行需要解决的问题是，根据数据集 D，建立一个信用等级分析模型，并根据这个模型，产生一系列规则。当银行在未来的某个时刻收到某个客户的贷款申请时，依据这些规则，可以根据该客户的年龄、职业、月薪等属性，来预测其信用等级，以确定是否提供贷款给该用户。这里的信用等级分析模型，就可以是一棵决策树。

表 3-1 信用记录表

姓名	年龄	职业	月薪	……	信用等级
张山	23	学生	1000	……	良
王刚	45	公务员	3000	……	优
李德海	38	职员	800	……	差
……	……	……	……	……	……
陈芳	50	教师	2600		良

在这个案例中，研究的重点是“信用等级”这个属性。给定一个信用等级未知的客户，要根据他/她的其他属性来估计“信用等级”的值是“优”、“良”还是“差”，也就是说，要把这个客户划分到信用等级为“优”、“良”、“差”这 3 个类别的某一个类别中去。这里把“信用等级”这个属性称为**“类标号属性”**。数据集 *D* 中“信用等级”属性的全部取值就构成了**类别集合**：Class={“优”，“良”，“差”}。

在决策树方法中，有两个基本的步骤。其一是构建决策树，其二是将决策树应用于数据库。大多数研究都集中在如何有效地构建决策树，而应用则相对比较简单。构建决策树的算法比较多，在 Clementine 中提供了 4 种算法，包括 C&RT、CHAID、QUEST 和 C5.0。采用其中的某种算法，输入训练数据集，就可以构造出一棵类似于图 3.1 所示的决策树。

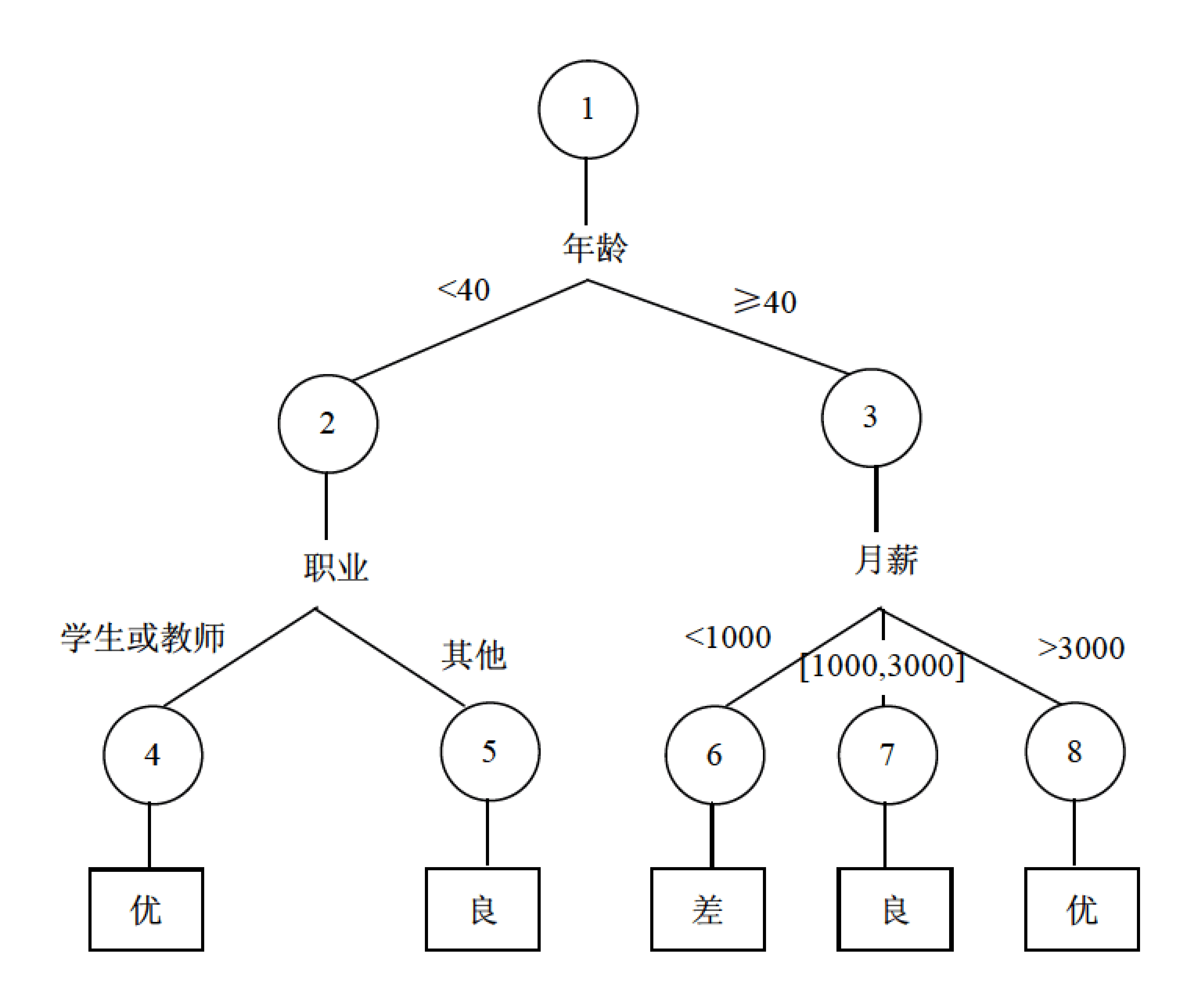


图 3.1 决策树举例

一棵决策树是一棵有向无环树，它由若干个**节点**、**分支**、**分裂谓词**以及**类别**组成。

节点是一棵决策树的主体。其中，没有父节点的节点称为**根节点**，如图 3.1 中的节点 1；没有子节点的节点称为**叶子节点**，如图 3.1 中的节点 4、5、6、7、8。一个节点按照某个属性被分裂时，这个属性称为**分裂属性**，如节点 1 按照“年龄”被分裂，这里“年龄”就是分裂属性，同理，“职业”、“月薪”也是分裂属性。每一个分支都会被标记一个

分裂谓词，这个分裂谓词就是分裂父节点的具体依据，例如在将节点 1 分裂时，产生两个分支，对应的分裂谓词分别是“年龄<40”和“年龄≥40”。另外，每一个叶子节点都被确定一个类标号，这里是“优”、“良”或者“差”。

实际上，每一个节点都代表了一个数据集。根节点 1 代表了初始数据集 D，其他节点都是数据集 D 的子集。例如，节点 2 代表数据集 D 中年龄小于 40 岁的那部分样本组成的数据集。子节点是父节点的子集。

基于以上描述，下面给出决策树的定义：

给定一个数据集 $D=\{t_1, t_2, \cdots, t_n\}$，其中 $t_i=<t_{i1}, t_{i2}, \cdots, t_{ih}>$是 D 中的第 i 个样本（i=1, 2, ⋯, n），数据集模式包含的属性集为$\{A_1, A_2, \cdots, A_h\}$，$t_{ij}$ 是第 i 个样本的第 j 个属性 A_j 的值（j=1, 2, ⋯, h）。同时给定类标号集合 $C=\{C_1, C_2, \cdots, C_m\}$。对于数据集 D，决策树是指具有下列 3 个性质的树：

- 每个非叶子节点都被标记一个分裂属性 A_i。
- 每个分支都被标记一个分裂谓词，这个分裂谓词是分裂父节点的具体依据。
- 每个叶子节点都被标记一个类标号 $C_j \in C$。

由此可以看出，构建一棵决策树，关键问题就在于，如何选择一个合适的分裂属性来进行一次分裂，以及如何制定合适的分裂谓词来产生相应的分支。各种决策树算法的主要区别也正在于此。

2. 修剪决策树

利用决策树算法构建了初始的树之后，为了有效地分类，还要对其进行**剪枝**。这是因为，由于数据表示不当、有噪声等原因，会造成生成的决策树过大或过度拟合。因此为了简化决策树，寻找一棵最优的决策树，剪枝是一个必不可少的过程。

通常，决策树越小，就越容易理解，其存储与传输的代价也就越小，但决策树过小会导致错误率较大。反之，决策树越复杂，节点越多，每个节点包含的训练样本个数越少，则支持每个节点的样本数量也越少，可能导致决策树在测试集上的分类错误率越大。因此，剪枝的基本原则就是，在保证一定的决策精度的前提下，使树的叶子节点最少，叶子节点的深度最小。要在树的大小和正确率之间寻找平衡点。

不同的算法，其剪枝的方法也不尽相同。常有的剪枝方法有**预剪枝**和**后剪枝**两种。例如 CHAID 和 C5.0 采用预剪枝，CART 则采用后剪枝。

预剪枝，是指在构建决策树之前，先制定好生长停止准则（例如指定某个评估参数的阈值），在树的生长过程中，一旦某个分支满足了停止准则，则停止该分支的生长，这样就可以限制树的过度生长。采用预剪枝的算法有可能过早地停止决策树的构建过程，但由于不必生成完整的决策树，算法的效率很高，适合应用于大规模问题。

后剪枝，是指待决策树完全生长结束后，再根据一定的规则，剪去决策树中那些不具一般代表性的叶子节点或者分支。这时，可以将数据集划分为两个部分，一个是训练数据集，一个是测试数据集。训练数据集用来生成决策树，而测试数据集用来对生成的决策树进行测试，并在测试的过程中通过剪枝来对决策树进行优化。

3. 生成规则

在生成一棵最优的决策树之后，就可以根据这棵决策树来生成一系列规则。这些规

则采用“If…，Then…”的形式。从根节点到叶子节点的每一条路径，都可以生成一条规则。这条路径上的分裂属性和分裂谓词形成规则的前件（If 部分），叶子节点的类标号形成规则的后件（Then 部分）。

例如，图 3.1 的决策树可以形成以下 5 条规则：

If (年龄<40) and (职业=“学生” or 职业=“教师”) Then 信用等级=“优”

If (年龄<40) and (职业!=“学生”and 职业!=“教师”) Then 信用等级=“良”

If (年龄≥40) and (月薪<1000) Then 信用等级=“差”

If (年龄≥40) and (月薪≥1000 and 月薪≤3000) Then 信用等级=“良”

If (年龄≥40) and (月薪>3000) Then 信用等级=“优”

这些规则即可应用到对未来观测样本的分类中了。

3.2 ID3、C4.5 与 C5.0

ID3 算法是最有影响力的决策树算法之一，由 Quinlan 提出。ID3 算法的某些弱点被改善之后得到了 C4.5 算法；C5.0 则进一步改进了 C4.5 算法，使其综合性能大幅度提高。但由于 C5.0 是 C4.5 的商业版本，其算法细节属于商业秘密，因此没有被公开，不过在许多数据挖掘软件包中都嵌入了 C5.0 算法，包括 Clementine。

3.2.1 ID3

1. 信息增益

任何一个决策树算法，其核心步骤都是为每一次分裂确定一个分裂属性，即究竟按照哪一个属性来把当前数据集划分为若干个子集，从而形成若干个“树枝”。

ID3 算法是采用“**信息增益**”为度量来选择分裂属性的。哪个属性在分裂中产生的信息增益最大，就选择该属性作为分裂属性。那么什么是信息增益呢？这需要首先了解“熵”这个概念。

熵，是数据集中的不确定性、突发性或随机性的程度的度量。当一个数据集中的记录全部都属于同一类的时候，则没有不确定性，这种情况下的熵就为 0。

决策树分裂的基本原则是，数据集被分裂为若干个子集后，要使每个子集中的数据尽可能的“纯”，也就是说子集中的记录要尽可能属于同一个类别。如果套用熵的概念，即要使分裂后各子集的熵尽可能的小。

例如在一次分裂中，数据集 D 被按照分裂属性“年龄”分裂为两个子集 D_1 和 D_2，如图 3.2 所示。

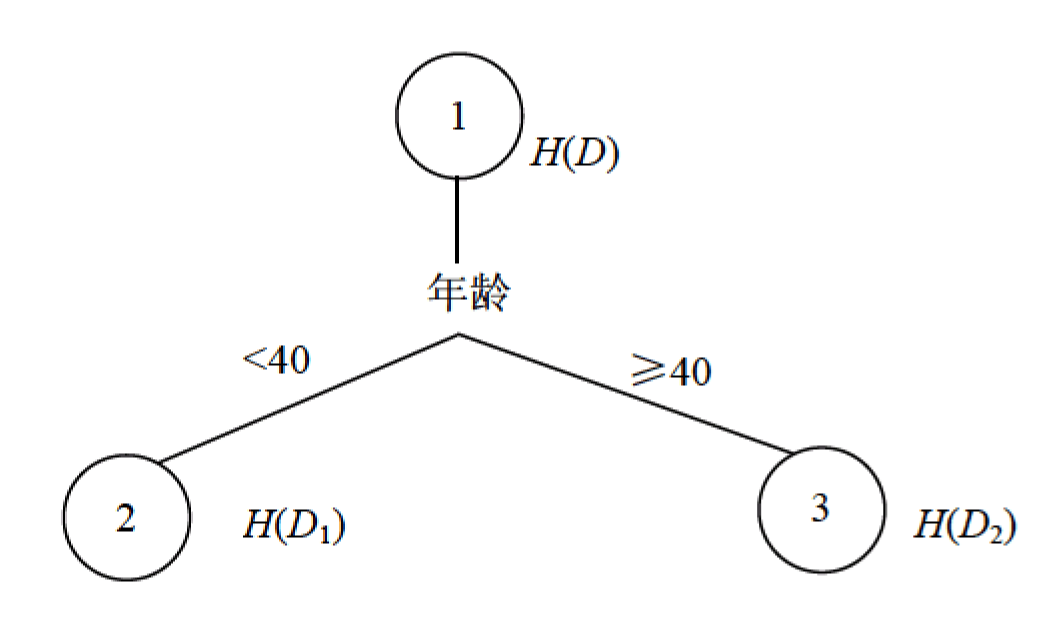


图 3.2 分裂中熵的变化

其中，$H(D)$、$H(D_1)$、$H(D_2)$分别是数据集 D、D_1、D_2 的熵。那么，按照“年龄”将数据集 D 分裂为 D_1 和 D_2 所得到的信息

增益为：

$$\text{Gain}(D, \text{年龄})= H(D)-[P(D_1)\times H(D_1)+ P(D_2)\times H(D_2)]$$

其中，$P(D_1)$是 D 中的样本被划分到 D_1 中的概率，$P(D_2)$是 D 中的样本被划分到 D_2 中的概率，$P(D_1)\times H(D_1)+ P(D_2)\times H(D_2)$是 $H(D_1)$和 $H(D_2)$的“加权和”。

显然，如果 D_1 和 D_2 中的数据越“纯”，$H(D_1)$和 $H(D_2)$就越小，信息增益就越大，或者说熵下降得越多。

按照这个方法，测试每一个属性的信息增益，选择增益值最大的属性作为分裂属性。

下面给出信息增益的计算方法。

设 S 是由 s 个样本组成的数据集。若 S 的类标号属性具有 m 个不同的取值，即定义了 m 个不同的类 C_i（i=1, 2, …, m）。设属于类 C_i 的样本的个数为 s_i，那么数据集 S 的熵为：

$$I(s_1, s_2, \cdots, s_m) = \sum_{i=1}^{m}\left[p_i \times \text{lb}\left(\frac{1}{p_i}\right)\right] = -\sum_{i=1}^{m}[p_i \times \text{lb}(p_i)]$$

其中，p_i 是任意样本属于类别 C_i 的概率，用 s_i/s 来估计。

设 S 的某个属性 A 具有 v 个不同值$\{a_1, a_2, \cdots, a_v\}$，先根据属性 A 将数据集 S 划分为 v 个不同的子集$\{S_1, S_2, \cdots, S_v\}$，子集 S_j 中所有样本的属性 A 的值都等于 a_j(j=1, 2, …, v)。

于是，在分裂为 v 个子集之后，任意一个子集 S_j 的熵为：

$$I(s_{1j}, s_{2j}, \cdots, s_{mj}) = -\sum_{i=1}^{m}[p_{ij} \times \text{lb}(p_{ij})]$$

其中，s_{ij}(i=1, 2, …, m)是子集 S_j 中属于类别 C_i 的样本数量，$p_{ij}=s_{ij}/|s_j|$是 S_j 中的样本属于类别 C_i 的概率，$|s_j|$是子集 S_j 中的样本数量。

所以，S 按照属性 A 划分出的 v 个子集的熵的加权和为：

$$E(S, A) = -\sum_{j=1}^{v}\left[\frac{s_{1j}+s_{2j}+\cdots+s_{mj}}{s} \times I(s_{1j}, s_{2j}, \cdots, s_{mj})\right]$$

其中，$\frac{s_{1j}+s_{2j}+\cdots+s_{mj}}{s}$ 是子集 S_j 的权重，等于子集 S_j 中的样本数量$|s_j|$除以 S 中的样本总数。

所以，按照属性 A 把数据集 S 分裂，所得的信息增益就等于数据集 S 的熵减去各子集的熵的加权和：

$$\text{Gain}(S, A) = I(s_1, s_2, \cdots, s_m) - E(S, A)$$

ID3 算法计算每一个属性的信息增益，并选取增益最大的属性作为分裂属性，然后用该属性的每一个不同的值创建一个分支，从而将数据集划分为若干个子集。

2. ID3 算法的流程

ID3 算法是一个从上到下、分而治之的归纳过程。

ID3 算法的核心是：在决策树各级节点上选择分裂属性时，通过计算信息增益来选择属性，以使得在每一个非叶节点进行测试时，能获得关于被测试样本最大的类别信息。其具体方法是：检测所有的属性，选择信息增益最大的属性产生决策树节点，由该属性

的不同取值建立分支，再对各分支的子集递归调用该方法建立决策树节点的分支，直到所有子集仅包含同一类别的数据为止。最后得到一棵决策树，它可以用来对新的样本进行分类。

ID3 算法思想描述如下：

（1）初始化决策树 T 为只含一个树根(X, Q)，其中 X 是全体样本集，Q 为全体属性集。

（2）if(T 中所有叶节点(X', Q')都满足 X 属于同一类或 Q' 为空) then 算法停止；

（3）else

{任取一个不具有（2）中所述状态的叶节点(X', Q')；

（4） for each Q' 中的属性 A do 计算信息增益 Gain(A, X')；

（5） 选择具有最高信息增益的属性 B 作为节点(X', Q')的分裂属性；

（6） for each B 的取值 b_i do

{从该节点(X', Q')伸出分支，代表分裂输出 $B=b_i$；

求得 X 中 B 值等于 b_i 的子集 X_i，并生成相应的叶节点$(X_i', Q', \{B\})$；}

（7）转（2）；}

下面通过一个实例来了解一下决策树的构建过程。

表 3-2 是一个假想的银行贷款客户历史信息（略去了客户姓名），包含 14 个样本。现要求以这 14 个样本为训练数据集，以“提供贷款”为类标号属性，用 ID3 算法构建决策树。

表 3-2 训练数据集 D

No.	年龄	收入水平	有固定收入	VIP	类别：提供贷款
1	<30	高	否	否	否
2	<30	高	否	是	否
3	[30，50]	高	否	否	是
4	>50	中	否	否	是
5	>50	低	是	否	是
6	>50	低	是	是	否
7	[30，50]	低	是	是	是
8	<30	中	否	否	否
9	<30	低	是	否	是
10	>50	中	是	否	是
11	<30	中	是	是	是
12	[30，50]	中	否	是	是
13	[30，50]	高	是	否	是
14	>50	中	否	是	否

设该数据集为 D。类标号属性“提供贷款”有两个不同值：“是”和“否”，因此有两个不同的类。令 C_1 对应“是”，C_2 对应“否”，那么 C_1 有 9 个样本，C_2 有 5 个样本，所以数据集 D 的熵为

$$I(s_1,s_2)=I(9,5)=-\frac{9}{14}\text{lb}\left(\frac{9}{14}\right)-\frac{5}{14}\text{lb}\left(\frac{5}{14}\right)=0.9406$$

下面分别计算按各属性分裂后所得诸子集的熵的加权和。

若以“年龄”作为分裂属性，则产生 3 个子集（因为该属性有 3 个不同的取值），所以 D 按照属性“年龄”划分出的 3 个子集的熵的加权和为

$$E(D,\text{年龄})=\frac{5}{14}\left(-\frac{3}{5}\text{lb}\frac{3}{5}-\frac{2}{5}\text{lb}\frac{2}{5}\right)+\frac{4}{14}\left(-\frac{4}{4}\text{lb}\frac{4}{4}\right)+\frac{5}{14}\left(-\frac{3}{5}\text{lb}\frac{3}{5}-\frac{2}{5}\text{lb}\frac{2}{5}\right)$$
$$=0.3468+0+0.3468=0.6936$$

因此，以“年龄”作为分裂属性来划分的信息增益为

$$\text{Gain}(D,\text{年龄})=I(s_1,s_2)-E(D,\text{年龄})=0.9406-0.6936=0.247$$

同理，若以“收入水平”为分裂属性：

$$E(D,\text{收入水平})=\frac{4}{14}\left(-\frac{2}{4}\text{lb}\frac{2}{4}-\frac{2}{4}\text{lb}\frac{2}{4}\right)+\frac{6}{14}\left(-\frac{4}{6}\text{lb}\frac{4}{6}-\frac{2}{6}\text{lb}\frac{2}{6}\right)$$
$$+\frac{4}{14}\left(-\frac{3}{4}\text{lb}\frac{3}{4}-\frac{1}{4}\text{lb}\frac{1}{4}\right)=0.2857+0.3936+0.2318=0.9111$$

$$\text{Gain}(D,\text{收入水平})=I(s_1,s_2)-E(D,\text{收入水平})=0.9406-0.9111=0.0295$$

若以“有固定收入”为分裂属性：

$$E(D,\text{固定收入})=\frac{7}{14}\left(-\frac{4}{7}\text{lb}\frac{4}{7}-\frac{3}{7}\text{lb}\frac{3}{7}\right)+\frac{7}{14}\left(-\frac{6}{7}\text{lb}\frac{6}{7}-\frac{1}{7}\text{lb}\frac{1}{7}\right)$$
$$=0.4927+0.2959=0.7886$$

$$\text{Gain}(D,\text{固定收入})=I(s_1,s_2)-E(D,\text{固定收入})=0.9406-0.7886=0.152$$

若以“VIP”为分裂属性：

$$E(D,\text{VIP})=\frac{8}{14}\left(-\frac{6}{8}\text{lb}\frac{6}{8}-\frac{2}{8}\text{lb}\frac{2}{8}\right)+\frac{6}{14}\left(-\frac{3}{6}\text{lb}\frac{3}{6}-\frac{3}{6}\text{lb}\frac{3}{6}\right)$$
$$=0.4636+0.4286=0.8922$$

$$\text{Gain}(D,\text{VIP})=I(s_1,s_2)-E(D,\text{VIP})=0.9406-0.8922=0.0484$$

由此可见，若以“年龄”作为分裂属性，所得信息增益最大。于是根据该属性的 3 个不同取值，将数据集 D 分裂为 3 个子集：D_1、D_2 和 D_3，如图 3.3 所示。

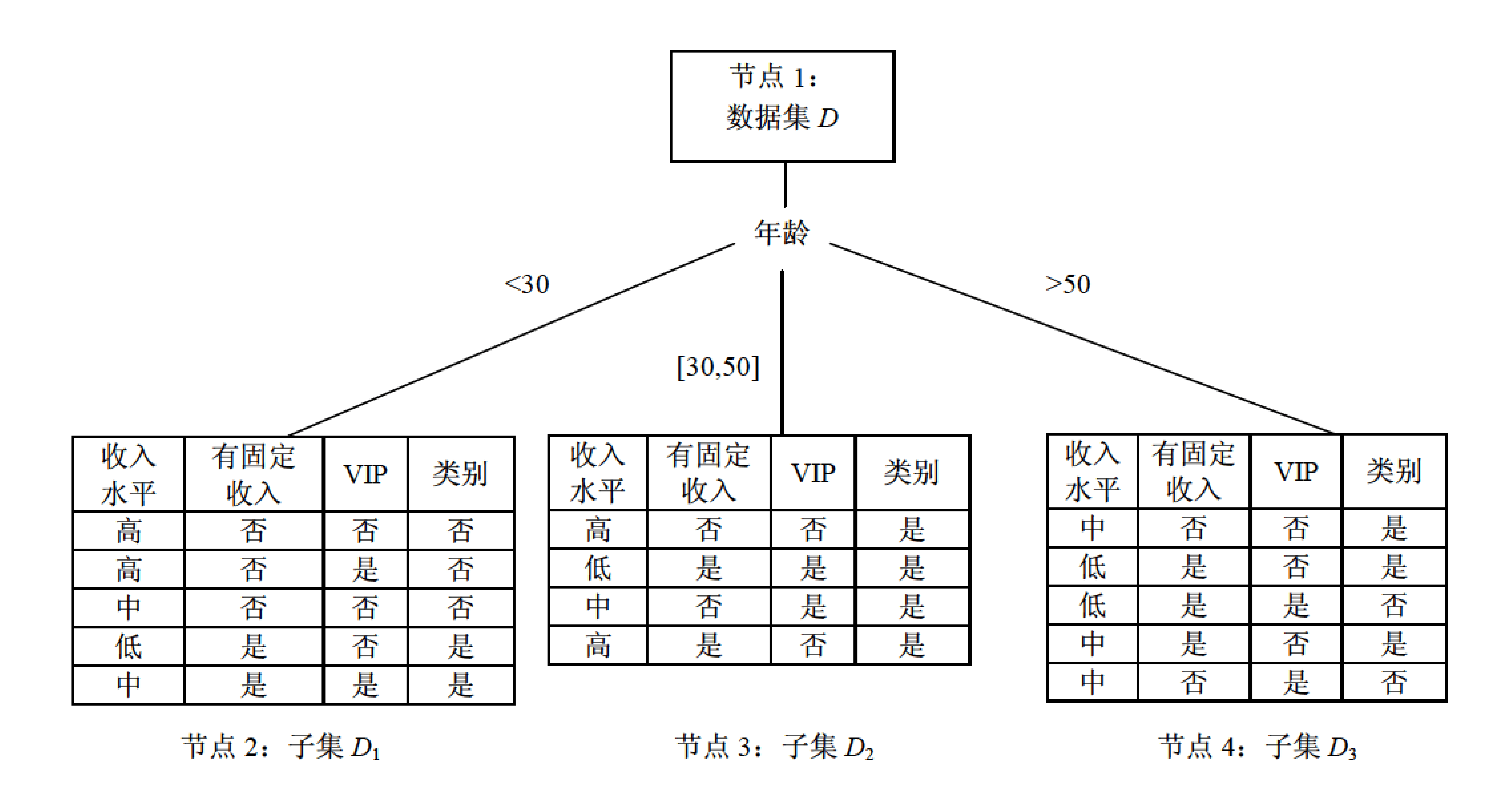
在图 3.3 中，节点 3 中的全部样本都属于同一个类别，因此它成为叶子节点，不再分裂。

采用同样的方法，分别对数据子集 D_1 和 D_3 进行分裂，直到所得子集的全部样本属于同一个类别，得到全部叶子节点。最终得到的决策树如图 3.4 所示。

3. ID3 算法的性能分析

ID3 算法是一种典型的决策树分类算法，后来发展的许多决策树算法都是以 ID3 算法为基础发展而来的。

ID3 算法的优点在于它构建决策树的速度比较快，它的计算时间随问题的难度只是线性地增加，适合处理大批量数据集。



收入水平	有固定收入	VIP	类别
高	否	否	否
高	否	是	否
中	否	否	否
低	是	否	是
中	是	是	是

收入水平	有固定收入	VIP	类别
高	否	否	是
低	是	是	是
中	否	是	是
高	是	否	是

收入水平	有固定收入	VIP	类别
中	否	否	是
低	是	否	是
低	是	是	否
中	是	否	是
中	否	是	否

图 3.3　分裂数据集

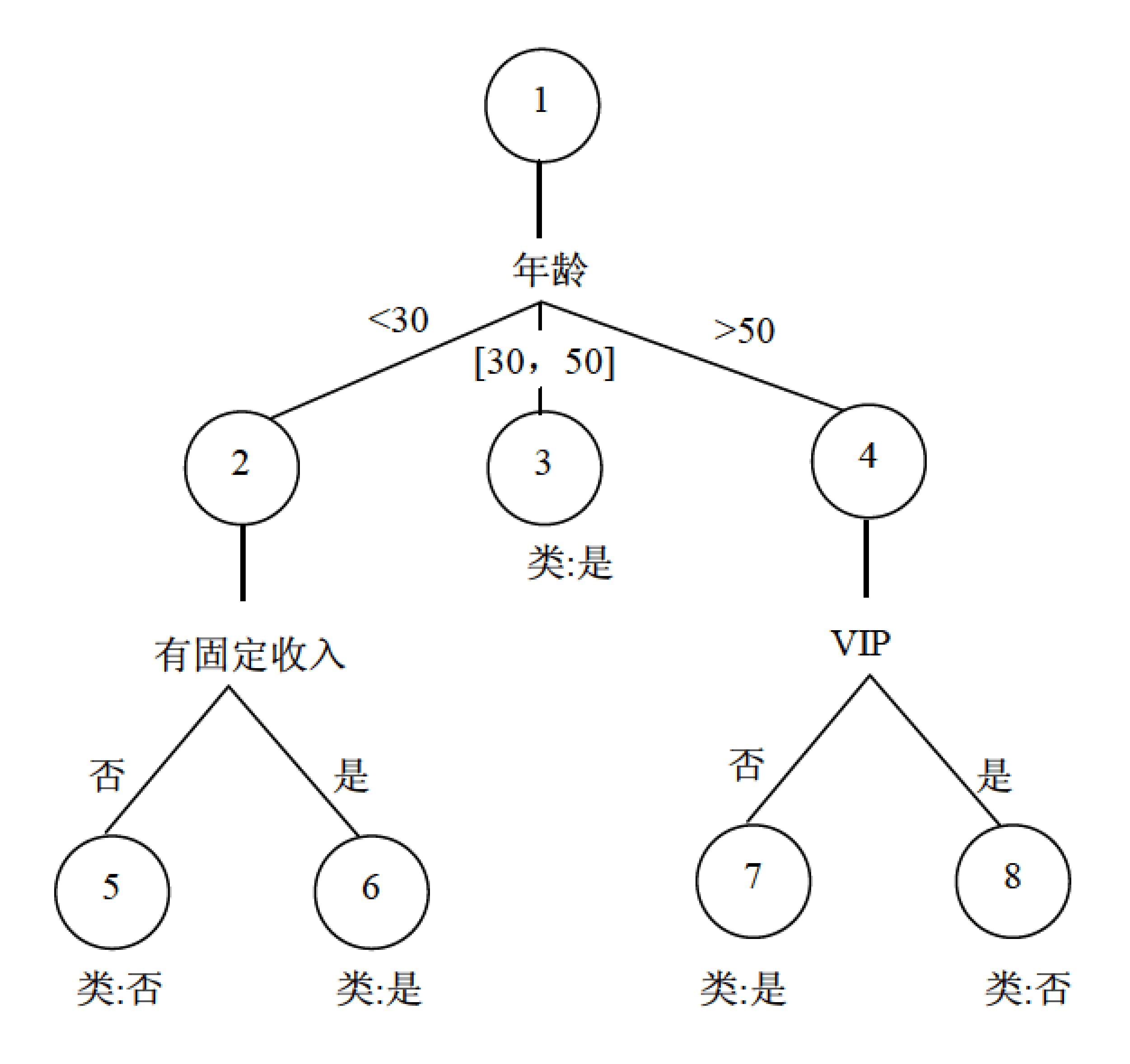


图 3.4　构建的决策树

同时，ID3 算法的缺点也是突出的，包括以下几点。

（1）ID3 算法对训练样本的质量的依赖性很强，训练样本的质量主要是指是否存在噪声和是否存在足够的样本。

（2）ID3 算法只能处理分类属性（离散属性），而不能处理连续属性（数值属性）。在处理连续属性时，一般要先将连续属性划分为多个区间，转化为分类属性。例如“年龄”，要把其数值事先转换为诸如“小于 30 岁”、“30～50 岁”、“大于 50 岁”这样的区间，再根据年龄值落入了某一个区间取相应的类别值。通常，区间端点的选取包含着一

定的主观因素和大量的计算。

（3）ID3 算法生成的决策树是一棵多叉树，分支的数量取决于分裂属性有多少个不同的取值。这不利于处理分裂属性取值数目较多的情况。因此目前流行的决策树算法大多采用二叉树模型。

（4）ID3 算法不包含对树的修剪，无法对决策树进行优化。

正因为 ID3 还存在着许多不足的地方，Quinlan 对 ID3 算法进行了改进，并于 1993 年提出了 ID3 的改进算法 C4.5。

3.2.2 C4.5

C4.5 算法的核心思想与 ID3 完全相同，但在实现方法上做了更好的改进，并增加了新的功能。主要包括：采用“增益比例”来选择分裂属性、对连续属性的处理、对样本属性值缺失情况的处理、规则的产生、交叉验证等。

1. 用“增益比例”来选择分裂属性

如前所述，ID3 是采用“信息增益”来选择分裂属性的。虽然这是一种有效的方法，但其具有明显的倾向性，即它倾向于选择具有大量不同取值的属性，从而产生许多小而纯的子集。尤其是关系数据库中作为主键的属性，每一个样本都有一个不同的取值。如果以这样的属性作为分裂属性，那么将产生非常多的分支，而且每一个分支产生的子集的熵均为 0（因为子集中只有一个样本！）。显然，这样的决策树是没有实际意义的。因此，Quinlan 提出使用增益比例来代替信息增益。

设 S 代表训练数据集，由 s 个样本组成。A 是 S 的某个属性，有 m 个不同的取值，根据这些取值可以把 S 划分为 m 个子集，S_i 表示第 i 个子集（i=1, 2, …, m），$|S_i|$表示子集 S_i 中的样本数量。那么：

$\text{Split_Info}(S,A)=-\sum_{i=1}^{m}\left(\frac{|S_i|}{s}\text{lb}\frac{|S_i|}{s}\right)$，称为**“数据集 S 关于属性 A 的熵”**。

$\text{Split_Info}(S,A)$ 用来衡量属性 A 分裂数据集的广度和均匀性。样本在属性 A 上的取值分布越均匀，$\text{Split_Info}(S,A)$ 的值就越大。

> **注意**
> 前面提到的“熵”，其实是“数据集关于类标号属性的熵”，也就是说，在计算熵的时候，样本个数是根据类别属性的值来计算的。这里“数据集 S 关于属性 A 的熵”，即样本个数是根据属性 A 的值来计算。

增益比例的定义为：

$\text{GainRatio}(S,A)=\frac{\text{Gain}(S,A)}{\text{Split_Info}(S,A)}$，C4.5 算法选择增益比例最大者作为分裂属性。

显然，属性 A 分裂数据集的广度越大，均匀性越强，其 $\text{Split_Info}(S,A)$ 就越大，增益比例就越小。所以，$\text{Split_Info}(S,A)$ 消除了选择那些值较多且均匀分布的属性作为分

裂属性的倾向性。

这里是一个计算增益比例的例子。对于表 3-2 所示的数据集 D，如果按照“年龄”来分裂，那么：

$$\text{Split_Info}(D,\text{年龄}) = -\frac{5}{14}\text{lb}\frac{5}{14} - \frac{4}{14}\text{lb}\frac{4}{14} - \frac{5}{14}\text{lb}\frac{5}{14} = 1.5774$$

所以，

$$\text{GainRatio}\,(D,\text{年龄}) = \frac{\text{Gain}(S,\text{年龄})}{\text{Split_Info}(S,\text{年龄})} = \frac{0.247}{1.5774} = 0.1566$$

同理可求出其他属性的信息增益比例，选择最大者作为分裂属性。

通常，C4.5 根据信息增益来选取分裂属性，对于超过平均信息增益的属性，才进一步根据增益比例来选取分裂属性。

2．连续属性的处理

ID3 最初的定义假设数据集的各属性都必须是离散的。如果有连续属性，则可以采用划分区间的方法来离散化。例如在前面的例子中，属性“年龄”由于是连续型属性，被划分为“<30”、“[30，50]”、“>50”3 个区间，这样属性“年龄”的不同取值就只有 3 个了，这就是被离散化的过程。在 C4.5 中，算法采用另外一种方法来实现连续属性的离散化。

设数据集中有一连续属性 Y，现要测试是否可以选用该属性来对数据集进行分裂以及如何分裂（即通过计算信息增益或增益比例来确认 Y 是否可以作为分裂属性，如果可以，还要确定分裂谓词）。

设属性 Y 有 m 个不同的取值，按大小顺序升序排列为 $v_1<v_2<$，…，$<v_m$。从 $\{v_1, v_2$，…，$v_{m-1}\}$ 中选择一个 v_i 作为阈值，则可以根据“$Y \leqslant v_i$”和“$Y > v_i$”将数据集划分为两个部分，形成两个分支。显然，$\{v_1, v_2$，…，$v_{m-1}\}$ 就是可能的阈值的集合，共 $(m-1)$ 个元素。把这些阈值一一取出来，并根据“$Y \leqslant v_i$”和“$Y > v_i$”把训练数据集划分为两个子集，并计算每一种划分方案下的信息增益或增益比例，选择最大增益或增益比例所对应的那个阈值，作为最优的阈值。

可以看出，如果选择连续属性作为分裂属性，则分裂后只有两个分支，而不像离散属性那样可能会有多个分支（由离散属性的取值个数决定）。另外，以上处理要付出一定的计算代价。

例如在表 3-3 所示的训练数据集中，如果要计算“年龄”属性的信息增益，则首先将不同的属性值排序{20，25，28，40，46，55，56，58，60，65，70}，那么可能的阈值集合为{20，25，28，40，46，55，56，58，60，65}，从中一一取出，并形成分裂谓词，例如取出“20”，形成谓词“≤20”和“>20”，用它们划分训练数据集，然后计算信息增益或增益比例。

3．处理有缺失值的样本

ID3 是基于所有属性值都已经确定这一假设的。但是在实际应用中，经常会因为搜集样本时有的样本的数据不完整，或者输入数据时有人为的误差等原因，一个数据集中

会有某些样本缺少一些属性值。例如在表 3-3 中，有两个样本的“收入水平”缺失了（用“？”代替）。

表 3-3　有连续属性和缺失值的训练数据集 S

No.	年龄	收入水平	有固定收入	VIP	类别：提供贷款
1	25	高	否	否	否
2	28	高	否	是	否
3	40	高	否	否	是
4	56	中	否	否	是
5	60	低	是	否	是
6	65	低	是	是	否
7	40	低	是	是	是
8	25	中	否	否	否
9	28	低	是	否	是
10	55	中	是	否	是
11	20	?	是	是	是
12	46	中	否	是	是
13	58	?	是	否	是
14	70	中	否	是	否

在用一个属性对数据集进行划分时，必须知道一个样本属于哪一类（以便于计算每类有多少个样本，进而计算该属性的信息增益），这是根据这个样本的属性值来决定的，但是由于属性值缺失，那么该如何判断这个样本属于哪一类呢？

C4.5 并不会武断地将一个有缺失值的样本抛弃（当然，样本数量很大的时候可以这么做），也不会随意地将它分配到某个类别中去。C4.5 会根据其他已知属性值来计算一个有缺失值的样本属于某个类别的概率，这个样本可以属于每一个类，只是概率不同而已。

例如，在表 3-3 的 14 个样本中，“收入水平”有两个缺失值，其他 12 个样本的分布如表 3-4 所示。

表 3-4　样本分布

收入水平	分类：是	分类：否	合计	收入水平	分类：是	分类：否	合计
高	1	2	3	低	3	1	4
中	3	2	5	合计	7	5	12

由表 3-4 可以看出，一个未知的“收入水平”的值，取为“高”的概率为 3/12，取为“中”的概率为 5/12，取为“低”的概率为 4/12。所以，当用属性“收入水平”将该数据集 S 划分为 3 个子集 S_1（收入水平=“高”）、S_2（收入水平=“中”）和 S_3（收入水平=“低”）后：

S_1 的样本数量为：3+2×(3/12)；S_2 的样本数量为：5+2×(5/12)；S_3 的样本数量为：4+2×(4/12)。

3 个子集的样本数量总和为：3+2×(3/12)+5+2×(5/12)+4+2×(4/12)=14，样本总数不变。

有了各子集的样本数量，就可以计算相应的熵和信息增益了。

4．树的修剪

C4.5 采用的修剪方法是所谓的“后剪枝”，即待决策树完全生长结束之后，再来修剪掉那些对分类精度贡献不大的叶子节点。

对于某个节点，计算该节点分裂之前的误分类损失（由于错误地预测了样本的类别而导致的损失）和分裂成子树之后的误分类损失，如果分裂后的误分类损失没有得到显著降低，就可以考虑修剪掉这棵子树。

在计算分类精度之前，用户可以自行定义各种误分类损失的权重，例如“A 类样本被误分类为 B 类导致的损失”比“B 类样本被误分类为 A 类导致的损失”要大得多，在这种情况下就可以通过设置误分类损失的权重来加以区分。

5．规则的产生

C4.5 提供了将决策树模型转换为 If-Then 规则的算法。规则存储于一个二维数组中，每一行代表一个规则。表的每一列代表样本的一个属性，列的值代表了属性的不同取值。例如，对于分类属性来说，0 和 1 分别代表取属性的第一个和第二个值，对于数值属性而言，0 和 1 分别代表“小于等于阈值”和“大于阈值”。如果列值为–1，则代表规则中不包含该属性。

6．交叉验证

分类是有监督学习，通过学习可以对未知的数据进行预测。在训练过程开始之前，将一部分数据保留下来，在训练之后，利用这部分数据对学习的结果进行验证，这种模型评估方法称为交叉验证。如果将这个“学习-验证”的过程重复 k 次，就称为 k 次迭代交叉验证。首先将所有训练数据平均分成 k 份，每次使用其中一份作为测试样本，其余的 k–1 份数据作为学习样本，然后选择平均分类精度最高的树作为最后的结果。通常，分类精度最高的树并不是节点最多的树。另外，交叉验证还可以用于决策树的修剪。k 次迭代交叉验证非常适合训练样本数目比较少的情形，但由于要构建 k 棵决策树，因此计算量非常大。

3.2.3 C5.0

C5.0 是 C4.5 的一个商业版本，被广泛应用于许多数据挖掘软件包中，如 Clementine，但它的精确算法并没有公开。C5.0 主要针对大数据集的分类。它的决策树归纳与 C4.5 很相近，但规则生成不同。

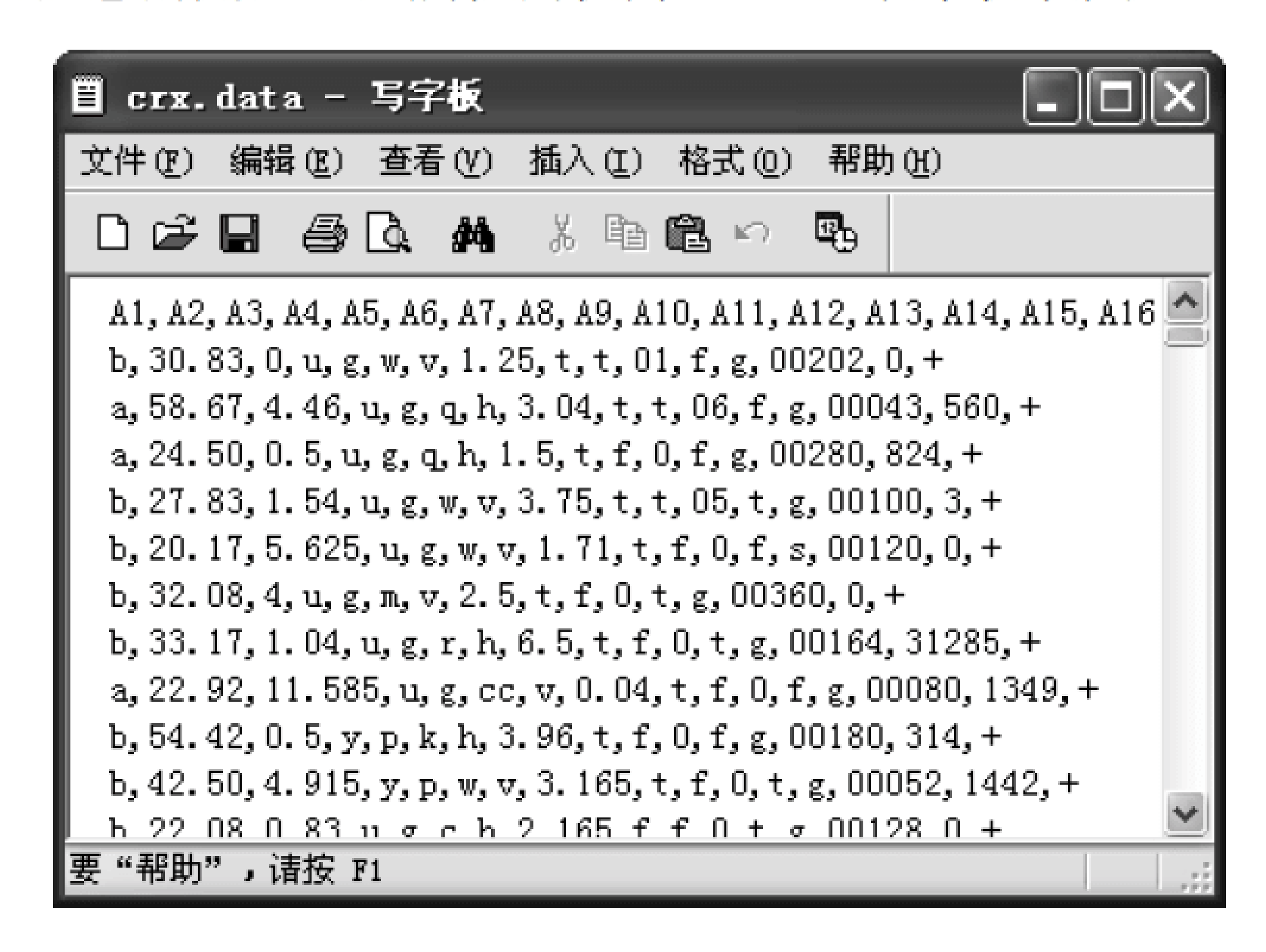
C5.0 包括了生成规则方面的改进。测试结果表明 C5.0 在内存占用方面的性能改善了大约 90%，在运行方面要比 C4.5 快 5.7~240 倍，并且生成的规则更加准确。

C5.0 在精度方面主要的改进源于采用了推进（boosting）方法。一些数据集上的测试结果表明，C5.0 的误差率比 C4.5 的一半还要低。

3.2.4 在 Clementine 中应用 C5.0

本节展示了一个在Clementine中用C5.0算法对某银行的信贷历史记录进行数据挖掘的案例，通过构建决策树并形成规则，为银行的信贷服务提供决策支持。目标是要找出银行批准或否决贷款人的信用申请的标准。

数据来自芝加哥大学尔湾分校知识发现数据档案库 UCI Knowledge Discovery in Databases Archive（ftp://ftp.ics.uci.edu/pub/machine-learning-databases/credit-screening/），存放在文件 crx.data 中。一共包含 690 个样本，16 个属性。由于商业保密的缘故，属性名称用 *A*1, *A*2, …, *A*16 来表示，其中 *A*16 是类标号属性，有两个取值：“+”和“–”，“+”表示信贷申请被通过，“–”表示信贷申请被否决。用写字板可以打开 crx.data 文件，如图 3.5 所示。需要注意的是，一些属性的值为“？”，表示值缺失。




```
A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16
b,30.83,0,u,g,w,v,1.25,t,t,01,f,g,00202,0,+
a,58.67,4.46,u,g,q,h,3.04,t,t,06,f,g,00043,560,+
a,24.50,0.5,u,g,q,h,1.5,t,f,0,f,g,00280,824,+
b,27.83,1.54,u,g,w,v,3.75,t,t,05,t,g,00100,3,+
b,20.17,5.625,u,g,w,v,1.71,t,f,0,f,s,00120,0,+
b,32.08,4,u,g,m,v,2.5,t,f,0,t,g,00360,0,+
b,33.17,1.04,u,g,r,h,6.5,t,f,0,t,g,00164,31285,+
a,22.92,11.585,u,g,cc,v,0.04,t,f,0,f,g,00080,1349,+
b,54.42,0.5,y,p,k,h,3.96,t,f,0,f,g,00180,314,+
b,42.50,4.915,y,p,w,v,3.165,t,f,0,t,g,00052,1442,+
```

图 3.5 原始数据文件

1. 获取数据

打开 Clementine，在选项板区单击“数据源”标签，可以看到所有的数据源节点，双击“可变文件”图标，即可将“可变文件”节点添加到数据流区域（如图 3.6 所示）。

“可变文件”节点可以用于从无格式文本文件中读取数据。右击数据流区域中的“可变文件”节点，在弹出的快捷菜单中选择“编辑”命令，弹出“可变文件”对话框，它包括 5 个标签：“文件”、“数据”、“过滤”、“类型”、“注解”。单击“文件”文本框后面的“浏览文件”按钮，找到并打开 crx.data 文件，即可获取数据，如图 3.7 所示。

其他选项保持默认设置。单击“注解”标签，将“名称”选项设置为“自定义”，并在其后的文本框中输入“crx.data 原始数据”，为这个数据源节点命名（如图 3.8 所示），然后单击“确定”按钮。

现在数据已经被读取到 Clementine 中，如果此时想浏览一下数据，可以向流中添加一个“表”节点：在选项板区单击“输出”标签，可以看到所有的输出节点，双击“表”，即可将“表”节点添加到数据流区域，并自动建立从“crx.data 原始数据”到“表”节点

的连接（如图 3.9 所示）。



图 3.6　添加数据源节点



图 3.7　编辑“数据源节点”

crx.data原始数据

刷新

D:\clementine\书\c5\crx.data

名称: ○自动 ◉自定义 crx.data原始数据

工具提示文本:

关键字:

由 Administrator 创建于 2009年8月7日 下午9:42。 ID: id6YJNQVVHBZ6

尚未保存此对象。

文件 数据 过滤 类型 注解

确定 取消 应用 重置

图 3.8 节点重命名

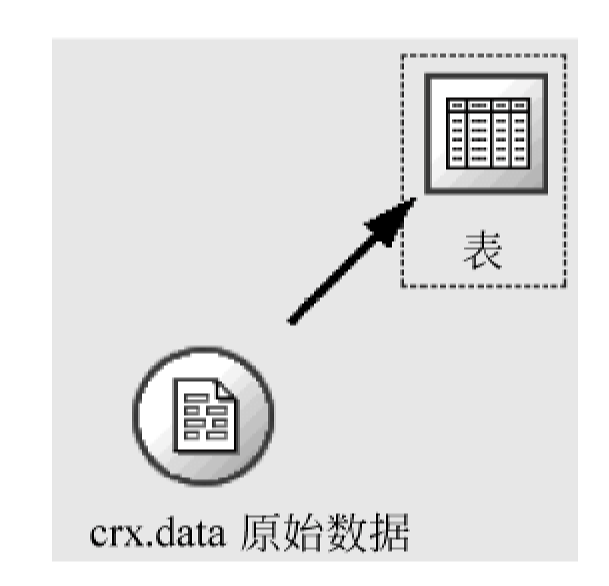


图 3.9 加入表节点

右击数据流区域中的“表”节点，在弹出的快捷菜单中选择“编辑”命令，弹出“表”对话框，单击 ▶ 执行(E) 按钮，即可浏览表中的数据（如图 3.10 所示），可以看到，共有 690 个样本。

表（16 个字段，690 条记录） #2

文件(F) 编辑(E) 生成(G)

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
675	a	37.330	2.500	u	g	i	h	0.210	f	f	0	f	g	260	246	-
676	a	41.580	1.040	u	g	aa	v	0.665	f	f	0	f	g	240	237	-
677	a	30.580	10.665	u	g	q	h	0.085	f	t	12	t	g	129	3	-
678	b	19.420	7.250	u	g	m	v	0.040	f	t	1	f	g	100	1	-
679	a	17.920	10.210	u	g	ff	ff	0.000	f	f	0	f	g	0	50	-
680	a	20.080	1.250	u	g	c	v	0.000	f	f	0	f	g	0	0	-
681	b	19.500	0.290	u	g	k	v	0.290	f	f	0	f	g	280	364	-
682	b	27.830	1.000	y	p	d	h	3.000	f	f	0	f	g	176	537	-
683	b	17.080	3.290	u	g	i	v	0.335	f	f	0	t	g	140	2	-
684	b	36.420	0.750	y	p	d	v	0.585	f	f	0	f	g	240	3	-
685	b	40.580	3.290	u	g	m	v	3.500	f	f	0	t	s	400	0	-
686	b	21.080	10.085	y	p	e	h	1.250	f	f	0	f	g	260	0	-
687	a	22.670	0.750	u	g	c	v	2.000	f	t	2	t	g	200	394	-
688	a	25.250	13.500	y	p	ff	ff	2.000	f	t	1	t	g	200	1	-
689	b	17.920	0.205	u	g	aa	v	0.040	f	f	0	f	g	280	750	-
690	b	35.000	3.375	u	g	c	h	8.290	f	f	0	t	g	0	0	-

表 注解

确定(O)

图 3.10 用“表”节点浏览数据

“表”节点用于根据数据建立一个表格，把数据显示在屏幕上，或者输出到一个文件中，供用户浏览。浏览之后，节点可以保留，也可以删除（选中节点后按 Delete 键）。

2. 处理缺失值

由于有缺失值的样本数量并不多，这里就简单地将这些样本丢弃。

双击“记录选项”标签下的“选择”节点，即可将节点添加到数据流区域，右击“crx.data 原始数据”节点，在快捷菜单中选择“连接”命令，然后单击“选择”节点，即可建立由“crx.data 原始数据”节点到“选择”节点的连接。右击“选择”节点，在快捷菜单中选择“编辑”命令，弹出“选择”对话框。在对话框中设置“模式”选项为“丢弃”，然后在“条件”框中输入：

*A*1="?" or @NULL(*A*2) or @NULL(*A*3) or *A*4="?" or *A*5="?" or *A*6="?" or *A*7="?" or @NULL(*A*8) or *A*9="?" or *A*10="?" or @NULL(*A*11) or *A*12="?" or *A*13="?" or @NULL(*A*14) or @NULL(*A*15) or *A*16="?"

这里，*A*1="?"是一个逻辑表达式，表示如果字符型属性 *A*1 的值为“？”，则该表达式的值为 True。@NULL(*A*2)表示如果数值型属性 *A*2 的值为空，则返回 True。

通过以上设置，即可将含有缺失值的样本全部丢弃。

在“选择”对话框的“注解”标签下，将“名称”选择为“自定义”，将节点命名为“过滤缺失值”，然后单击“确定”按钮。现在的数据流如图 3.11 所示。

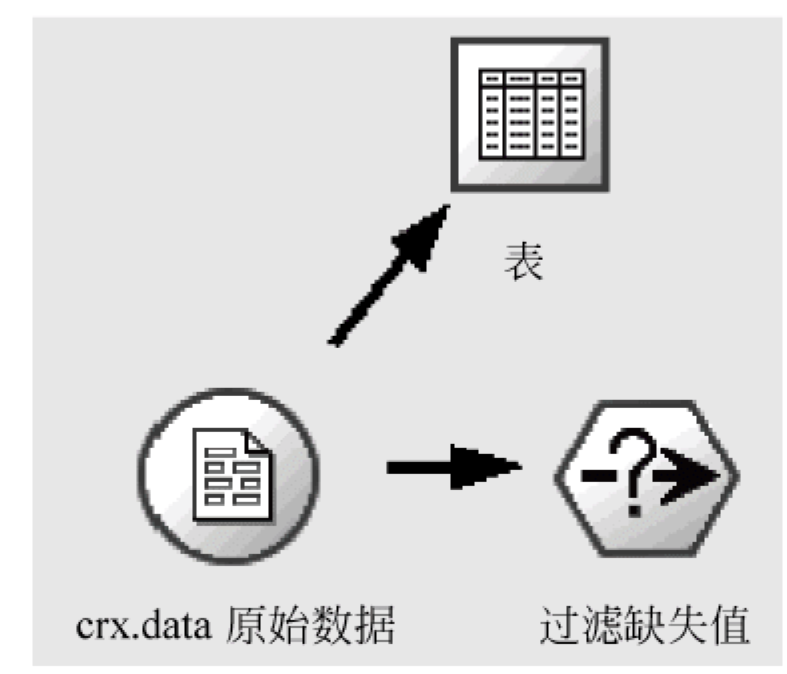


图 3.11 用选择节点处理缺失值

如果想查看丢弃缺失值样本后的数据，也可以像前面提到的方法一样，添加“表”节点，并将“过滤缺失值”节点与“表”节点连接，然后执行“表”节点，即可查看到有 653 个样本，即删除了 37 个有缺失值的样本。

3. 设置输出字段

对于决策树算法而言，其目的是根据一些非类标号属性的值，来预测类标号属性的值。所以，应该把这些非类标号属性设置为输入字段，类标号属性设置为输出字段。

设置输出字段，需要用到“类型”节点。双击“字段选项”标签下的“类型”节点，即可将“类型”节点添加到数据流区域，并建立与“过滤缺失值”节点的连接。然后设置“类型”节点的属性，如图 3.12 所示。

单击“读取值”按钮，即可显示各字段的类型和取值集合。可以看到主要有 3 种数据类型：“范围”型用于表示数值；“集”型用于表示多个具体值；“标志”型用于表示两个具体值。

最后，将类标号属性 *A*16 的方向设置为“输出”，单击“确定”按钮。

4. 设置训练数据集

到目前为止，数据集中共有 653 个样本。选择一半的样本出来，作为训练数据集，用于构建决策树模型。剩下的一半样本作为测试数据集，用生成的决策树进行分类预测，以评估决策树的性能。

从数据集中抽取部分样本出来，这就需要用到“抽样”节点。双击“记录选项”标签下的“抽样”节点，即可将“抽样”节点添加到数据流区域，并建立与“类型”节

点的连接。然后设置“抽样”节点的属性，如图 3.13 所示。



图 3.12　设置分类节点的属性



图 3.13　设置抽样属性

“采用方法”选择“简单”，“模式”选项设置为“包括样本”，“样本”选项设置为“第一个”，并在文本框中输入“327”，即选择前一半的样本。

在“注解”标签中，把该节点重命名为“训练数据集”，然后单击“确定”按钮。此时的数据流如图 3.14 所示。同样，如果想浏览训练数据集的数据，可以在“训练数据集”

后添加“表”节点并执行，可以看到只有 327 条记录。

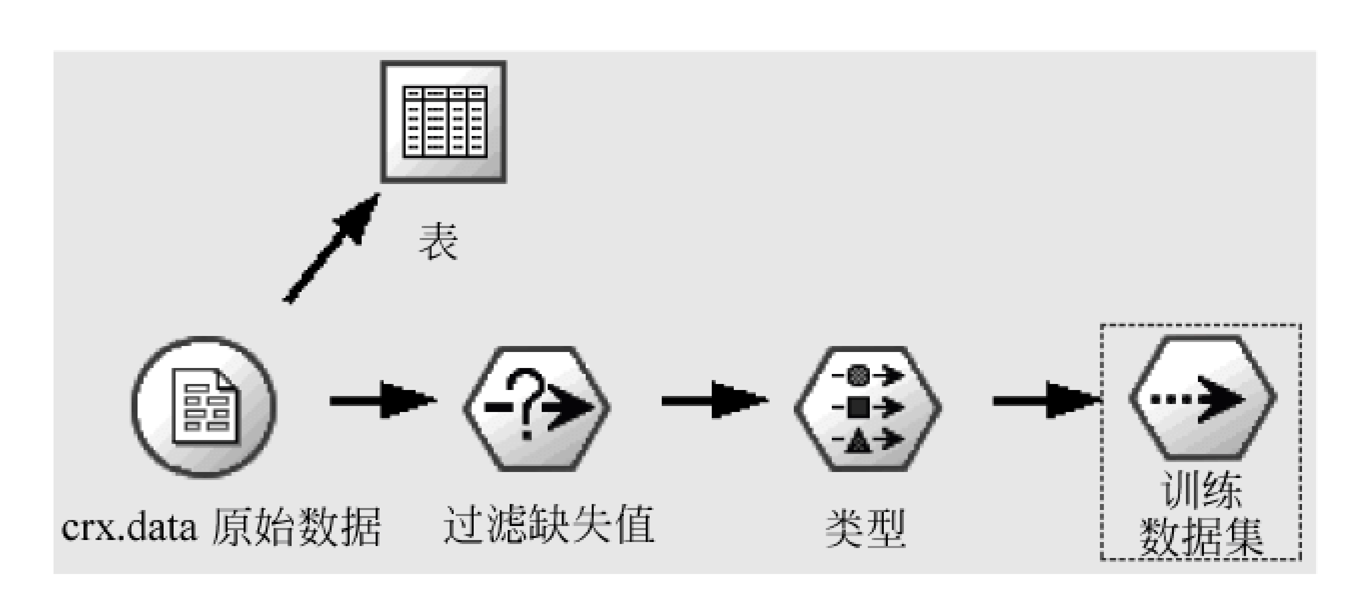


图 3.14　数据流

5．构建决策树模型

这是整个案例过程中的核心步骤，用于根据对训练数据集的学习来构建决策树。这需要用到 C5.0 节点。该节点使用 C5.0 算法生成决策树或者规则集。C5.0 模型根据能够带来最大信息增益的字段拆分样本。第一次拆分确定的样本子集随后再次拆分，通常是根据另一个字段进行拆分，这一过程重复进行直到样本子集不能再被拆分为止。最后，重新检验最低层次的拆分，那些对模型值没有显著贡献的样本子集将被剔除或者修剪。

双击“建模”标签下的 C5.0 节点，即可将 C5.0 节点添加到数据流区域，并自动命名为“A16”，同时建立与“训练数据集”节点的连接。双击“A16”节点即可打开其属性设置窗口，如图 3.15 所示。



图 3.15　设置 C5.0 节点的属性

使用分区数据（Use Partitioned Data）：如果用户定义了分割数据集，选择训练数据集作为建模数据集，并利用测试数据集对模型评价。

输出类型：有两个选项，即决策树和规则集。C5.0 可以生成这两种模型。决策树是对这种算法的拆分的直观描述。每一个叶子节点描述了训练数据的一个特定子集，而训练数据集中的每一种情况恰好属于树上的一个叶子节点。换句话说，决策树展示的任一特定数据记录只有一种可能预测。规则集则是规则的集合，试图对单个的记录做出预测。规则集从决策树中推出，从某种意义上说，以一种简化或者提炼的方式陈述决策树中的信息。这里选择“决策树”。

组符号（Group Symbolics）：如果选择了该选项，C5.0 会尝试将所有与输出字段格式相似的字符值合并。如果没有选择该选项，C5.0 会为用于拆分父节点的字符字段的每个值创建一个子节点。例如，如果 C5.0 按 COLOR 字段（包括 RED、GREEN 和 BLUE 3 个值）拆分，则默认创建三项拆分。但是，如果选择了该选项，并且 COLOR=RED 的记录与 COLOR=BLUE 的记录非常相似，则将创建二项拆分，COLOR 为 GREEN 的记录被分成一组，而 COLOR 为 RED 和 BLUE 的记录合为一组。

使用推进：C5.0 算法使用被称做推进（boosting）的方法提高其精确率。这种方法按序列建立多重模型。第一个模型以通常的方式建立。随后，建立第二个模型，聚焦于被第一个模型错误分类的记录。然后第三个模型聚焦于第二个模型的错误，等等。最后，应用整个模型集对样本进行分类，使用加权投票过程把分散的预测合并成综合预测。助推可以显著提高 C5.0 模型的精确度，但是同时也需要更长的训练时间。实验次数（The Number of Trials）选项允许控制用于助推的模型数量。

交叉验证（Cross-validate）：如果选择了该选项，C5.0 将使用一组基于训练数据子集建立的模型，来估计基于全部数据建立的模型的精确度。如果数据集过小，不能拆分成传统意义上的训练集和测试集，这将非常有用。在计算了精确度估计值后，用于交叉验证的模型将被丢弃。可以指定倍数，或用于交叉验证的模型数目。

模式：有两个选项，即“简单”和“专家”。对于简单的训练，可选择“简单”模式，则绝大多数 C5.0 参数自动设置。若选择“专家”模式，则可以设置“修剪严重性”和“每个子分支的最小记录数”。其中，**“修剪严重性”**（也称修剪纯度，Pruning Severity）决定生成决策树或规则集被修剪的程度，提高纯度值将获得更小、更简洁的决策树；降低纯度值将获得更加精确的决策树；这里设置为 40。**“每个子分支的最小记录数”**（Minimum Records Per Child Branch）用于限制决策树任一分支的拆分数。只有当两个或以上的后序子分支包括来自训练集的记录不少于最小记录数时，决策树才会继续拆分。默认值为 2，提高该值将有助于避免噪声数据的过度训练，这里设置为 20。

使用全局修剪（Use Global Pruning）。分两个阶段修剪树：第一个阶段是本地修剪，此时将检查子树并折叠分支以提高模型的准确性。第二个阶段是全局修剪，在此阶段中将把树视做一个整体并折叠虚弱的子树。默认情况下将执行全局修剪。要忽略全局修剪阶段，需要取消选中此选项。

辨别属性（**Winnow Attributes**）。如果选中此选项，C5.0 将在开始构建模型之前检

查预测变量的有效性。如果发现不相关的预测变量，则会将其从模型构建过程中排除。此选项对于具有许多预测变量字段的模型非常有用，并且有助于防止过度拟合。

单击对话框的“成本”标签，设置误分类损失矩阵，如图 3.16 所示。

在某些情况下，特定类型的错误比其他类错误所引起的损失更大。例如，把高风险信用卡申请者归入低风险信用类（一种错误）比把低风险信用卡申请者归入高风险类（另一种错误）损失要大。误分类损失允许指定不同类型预测错误之间的相对重要性。

误分类损失矩阵显示预测类和实际类每一可能组合的损失。所有的误分类损失都默认设置为 1.0。要输入自定义损失值，需要选择“使用误分类损失”（Use misclassification costs），然后把自定义值输入到损失矩阵中。当对树进行修剪时，在计算误分类损失的过程中，将把这些自定义损失值作为权重来影响误分类损失的计算结果。

在本案例中，对银行而言，拒绝“好”的客户和接受“坏”的客户这两种错误造成的损失并不相等。很显然，接受“坏”的客户，银行可能会遭受较大的违约风险；而拒绝“好”的客户，可能的损失则是贷款利息，或银行对客户作进一步了解所发生的较小费用。所以这里勾选“使用误分类损失”复选框，假设接受“坏”客户的成本是拒绝“好”客户成本的 2 倍。

单击“执行”按钮，即可生成决策树模型，并显示在管理器窗口的“模型”标签下，如图 3.17 所示。



图 3.16　误分类损失矩阵

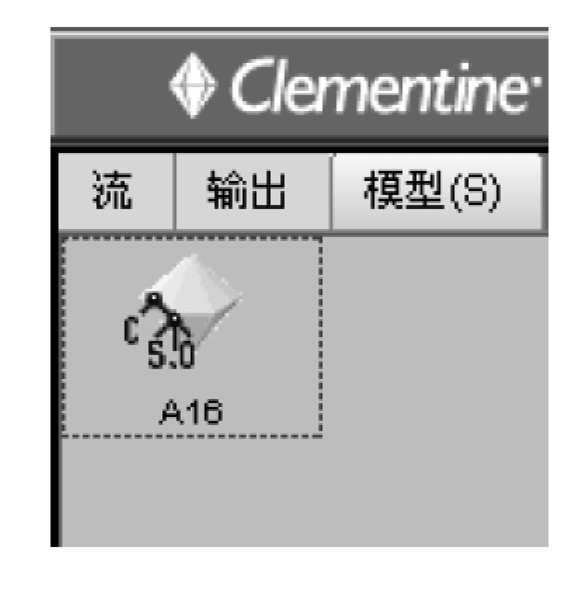


图 3.17　生成的决策树模型

右击该模型，在快捷菜单中选择“浏览”命令，可以查看生成的决策树，如图 3.18 所示。

在图 3.18 中，左侧以层的形式显示决策树，右侧显示的是每个变量的相对重要性。通常要将建模的主要精力放在最重要的变量上，并考虑丢弃和删除那些最不重要的变量。该图表指示了估计模型时各个变量的相对重要性。由于这些值都是相对值，因此显示的所有变量值总和为 1.0。变量重要性与模型准确性无关。需要注意的是，在 Clementine 建立的许多模型中都会显示变量重要性，但这里的变量重要性只是为我们提供关于变量

重要程度的参考信息，与模型的应用无关。变量重要性的计算方法，是从敏感性分析方法发展而来的，详细阐述请见相关文献（A. Saltelli，Stefano Tarantola，Francesca Campolongo，Marco Ratto.Sensitivity Analysis in Practice—A Guide to Assessing Scientific Models. 2004）。

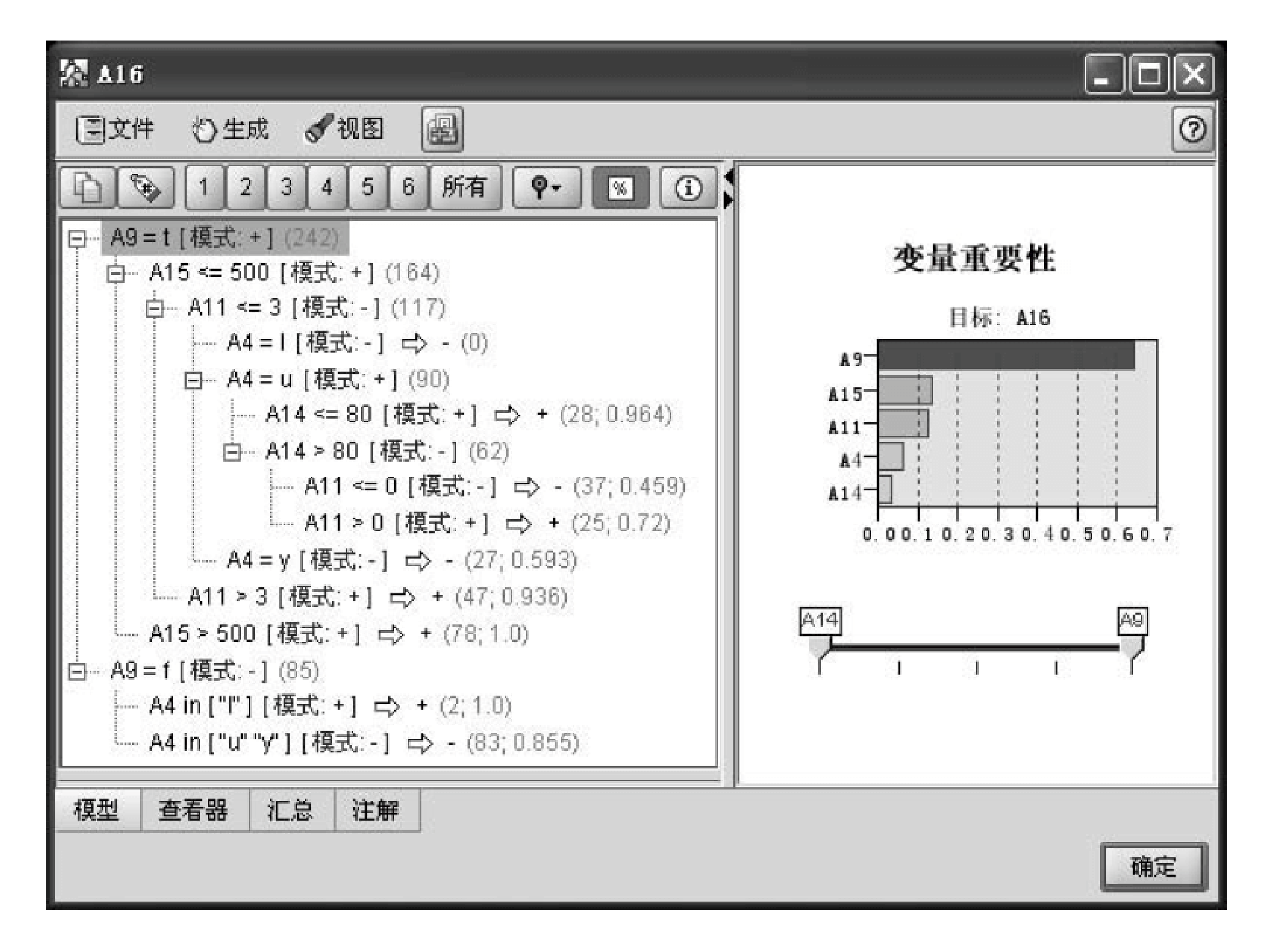


图 3.18　以层的形式显示决策树

单击“查看器”标签，则可以树的形式来显示决策树，显示的方式也可以选择横排或竖排，如图 3.19 所示。

6. 将测试数据集输入模型

为了评估生成的决策树模型的性能，我们把训练数据集输入决策树模型，用决策树模型根据训练数据集中各样本的非类标号属性的值来预测类标号属性的值，并将预测值和实际值相比较，即可了解分类预测的准确性。

首先，把数据集中除 327 个训练样本之外的 326 个样本取出，作为测试样本。单击“记录选项”标签下的“抽样”节点，然后单击数据流区域即可将“抽样”节点添加到数据流区域（或者将“抽样”节点拖入数据流区域），建立从“类型”节点到此抽样节点的连接。最后设置此抽样节点的属性，如图 3.20 所示。

“模式”选择“丢弃”，“样本”选择“从第一条记录开始连续抽取 327”，即把前 327 个样本丢弃，得到剩下 326 个样本，作为测试样本。

然后在“注解”标签下，为该抽样节点命名为“测试数据集”，单击“确定”按钮。

把生成的决策树模型从管理器窗口的“模型”标签下拖入数据流区域，并建立从“测试数据集”节点到决策树模型节点的连接。可以理解为，把测试数据集输入到决策树模型中，为了浏览决策树的输出结果，可以将结果输出到一个“表”节点，然后执行这个“表”节点。数据流的结果如图 3.21 所示。

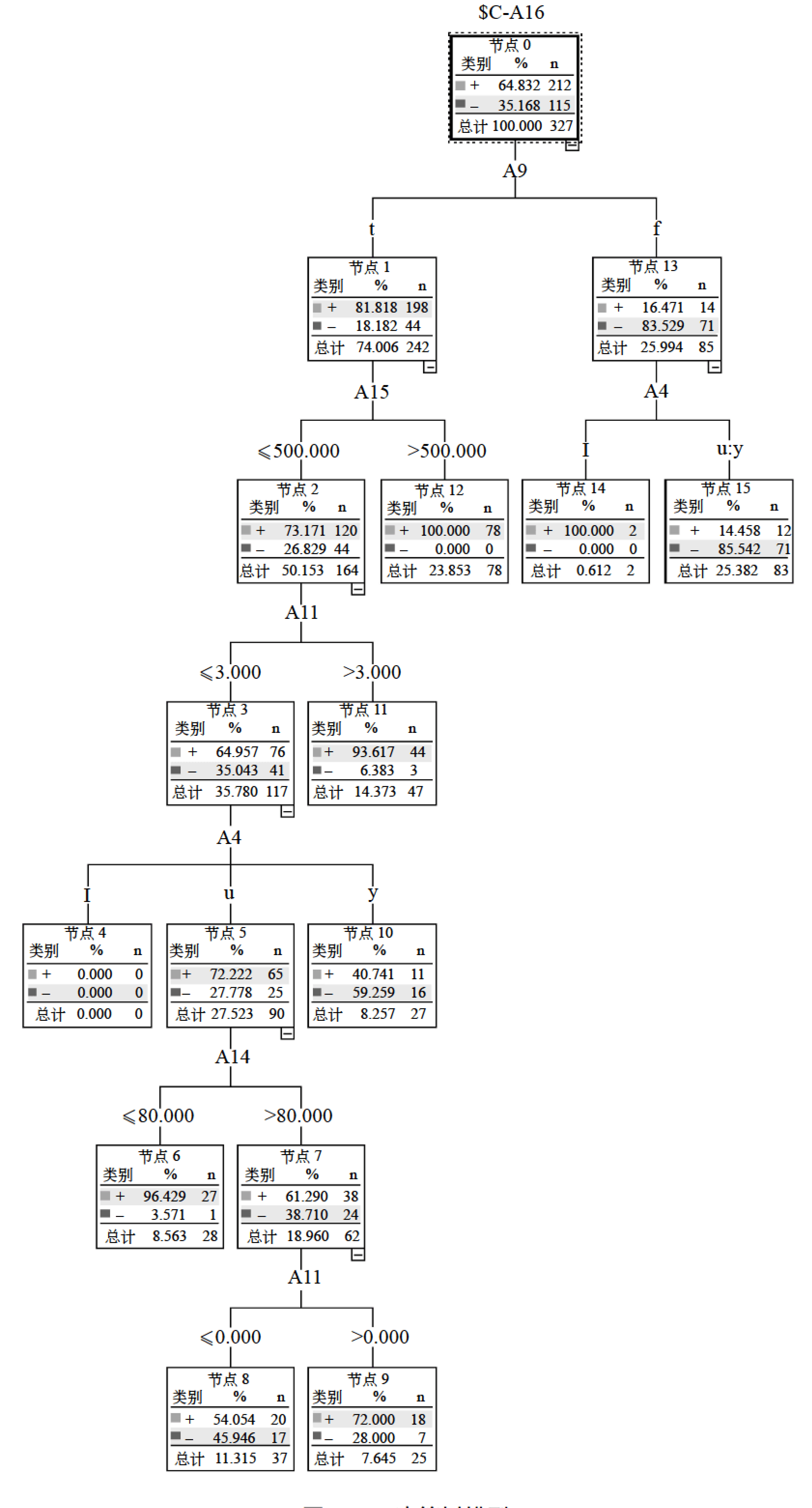

$C-A16
节点 0
类别 % n
+ 64.832 212
– 35.168 115
总计 100.000 327
A9
t
f
节点 1
类别 % n
+ 81.818 198
– 18.182 44
总计 74.006 242
节点 13
类别 % n
+ 16.471 14
– 83.529 71
总计 25.994 85
A15
A4
⩽500.000
>500.000
l
u:y
节点 2
类别 % n
+ 73.171 120
– 26.829 44
总计 50.153 164
节点 12
类别 % n
+ 100.000 78
– 0.000 0
总计 23.853 78
节点 14
类别 % n
+ 100.000 2
– 0.000 0
总计 0.612 2
节点 15
类别 % n
+ 14.458 12
– 85.542 71
总计 25.382 83
A11
⩽3.000
>3.000
节点 3
类别 % n
+ 64.957 76
– 35.043 41
总计 35.780 117
节点 11
类别 % n
+ 93.617 44
– 6.383 3
总计 14.373 47
A4
l
u
y
节点 4
类别 % n
+ 0.000 0
– 0.000 0
总计 0.000 0
节点 5
类别 % n
+ 72.222 65
– 27.778 25
总计 27.523 90
节点 10
类别 % n
+ 40.741 11
– 59.259 16
总计 8.257 27
A14
⩽80.000
>80.000
节点 6
类别 % n
+ 96.429 27
– 3.571 1
总计 8.563 28
节点 7
类别 % n
+ 61.290 38
– 38.710 24
总计 18.960 62
A11
⩽0.000
>0.000
节点 8
类别 % n
+ 54.054 20
– 45.946 17
总计 11.315 37
节点 9
类别 % n
+ 72.000 18
– 28.000 7
总计 7.645 25

图 3.19　决策树模型



图 3.20　抽取测试数据集

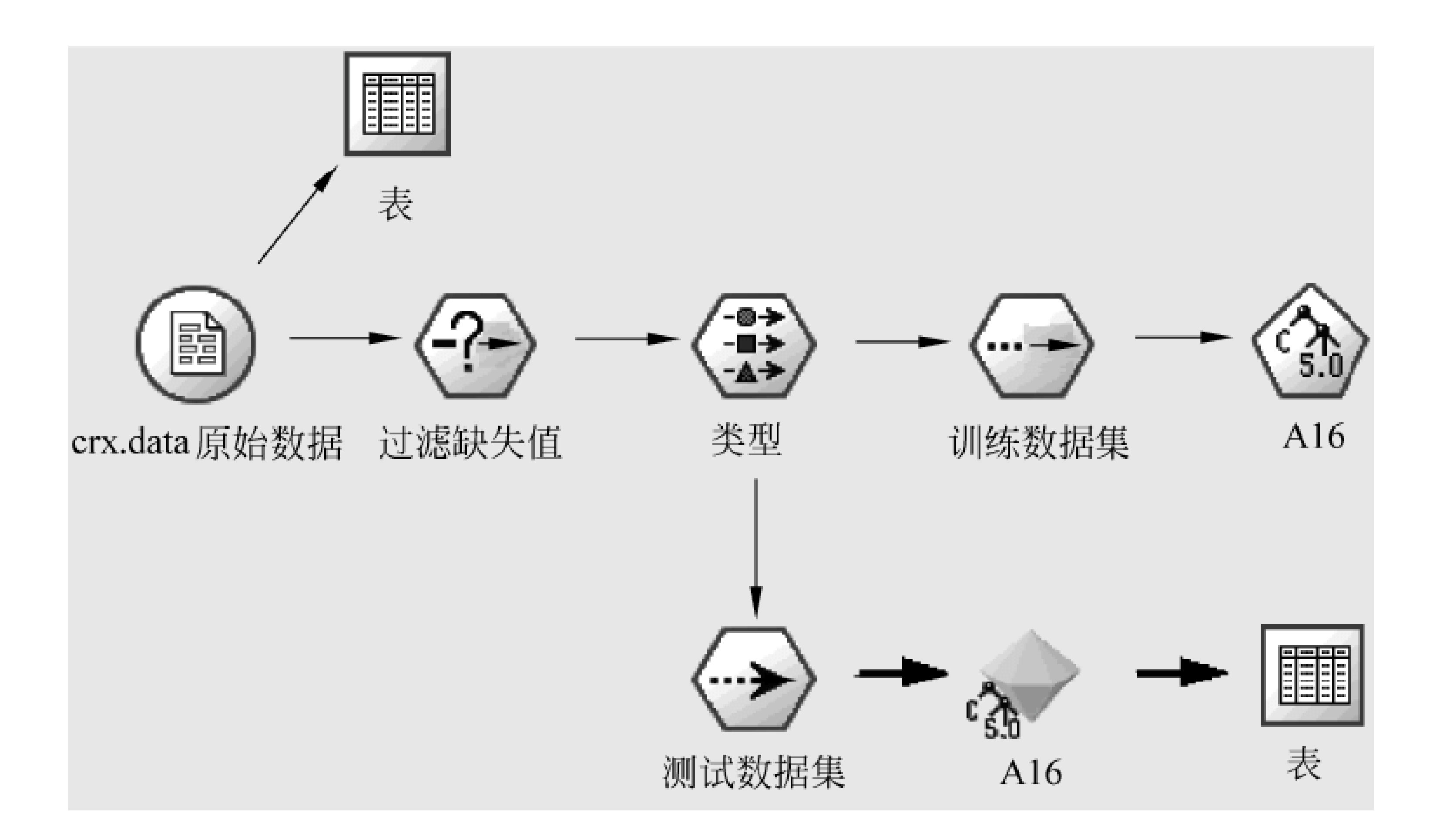
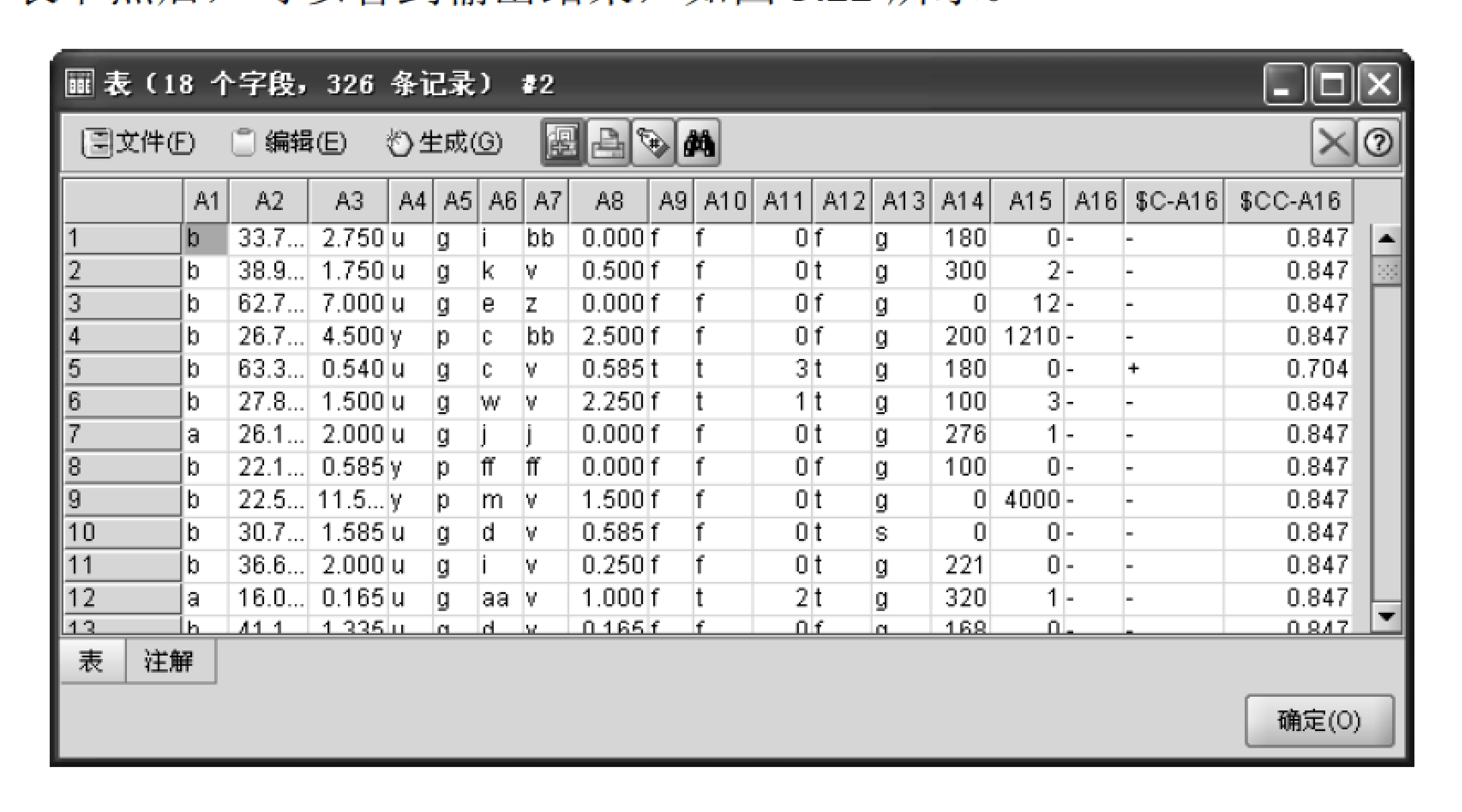


图 3.21　数据流

执行表节点后，可以看到输出结果，如图 3.22 所示。



	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	$C-A16	$CC-A16
1	b	33.7...	2.750	u	g	i	bb	0.000	f	f	0	f	g	180	0	-	-	0.847
2	b	38.9...	1.750	u	g	k	v	0.500	f	f	0	t	g	300	2	-	-	0.847
3	b	62.7...	7.000	u	g	e	z	0.000	f	f	0	f	g	0	12	-	-	0.847
4	b	26.7...	4.500	y	p	c	bb	2.500	f	f	0	f	g	200	1210	-	-	0.847
5	b	63.3...	0.540	u	g	c	v	0.585	t	t	3	t	g	180	0	-	+	0.704
6	b	27.8...	1.500	u	g	w	v	2.250	f	t	1	t	g	100	3	-	-	0.847
7	a	26.1...	2.000	u	g	j	j	0.000	f	f	0	t	g	276	1	-	-	0.847
8	b	22.1...	0.585	y	p	ff	ff	0.000	f	f	0	f	g	100	0	-	-	0.847
9	b	22.5...	11.5...	y	p	m	v	1.500	f	f	0	t	g	0	4000	-	-	0.847
10	b	30.7...	1.585	u	g	d	v	0.585	f	f	0	t	s	0	0	-	-	0.847
11	b	36.6...	2.000	u	g	i	v	0.250	f	f	0	t	g	221	0	-	-	0.847
12	a	16.0...	0.165	u	g	aa	v	1.000	f	t	2	t	g	320	1	-	-	0.847
13	b	41.1...	1.335	u	g	d	v	0.165	f	f	0	f	g	168	0	-	-	0.847

图 3.22　决策树的输出结果

可以看到，在原有 16 个属性基础上，又多了两个属性“$C-A16”、“$CC-A16”。其中，$C-A16 是模型的预测值（A16 是实际值），$CC-A16 是该预测值的置信度。

7．对预测结果的分析

对预测结果进行分析，需要用到“分析”节点。将“分析”节点（在“输出”标签下）拖入到数据流区域，并建立从生成决策树节点 A16 到“分析”节点的连接（如图 3.23 所示）。

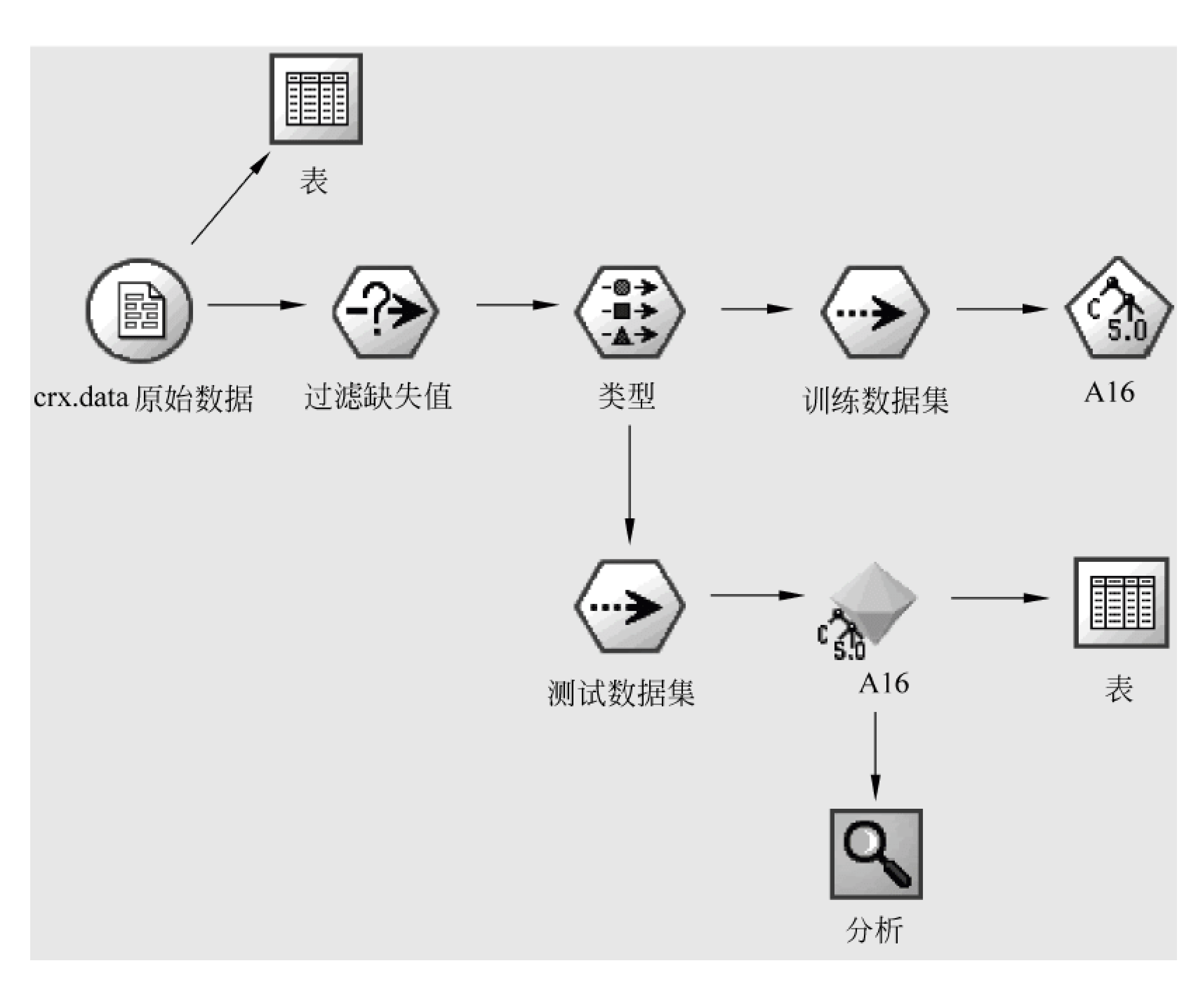


图 3.23　连接到“分析”节点的数据流

编辑“分析”节点的属性，如图 3.24 所示。

符合矩阵：显示对于符号型目标变量的每个被生成的（被预测的）字段和它的目标字段之间匹配的模式。用一个表格来显示，它的行被定义为实际值，列被定义为预测值，每一个单元格里是模式的记录数。

绩效评价：对符号型输出的模型显示绩效评价统计量。这些统计量报告输出字段的每一类别，是一种平均信息量的度量。

置信数字：对于生成一个置信度字段的模型，这个选项报告通过置信值上的统计量和它们的关系来预测。对于这个选项有两个设置：**阈值**，报告在指定为百分数的精确度以上的置信水平；**改善准确性**，报告这样的置信水平，在此水平之上的精度是由指定的因子提高的。例如，若全部的精确度是 90%并且这个选项被设置为 2.0，这个被报告的值将变为所要求的 95%精确度的置信度。

单击“执行”按钮后，即可看到分析的结果，如图 3.25 所示。



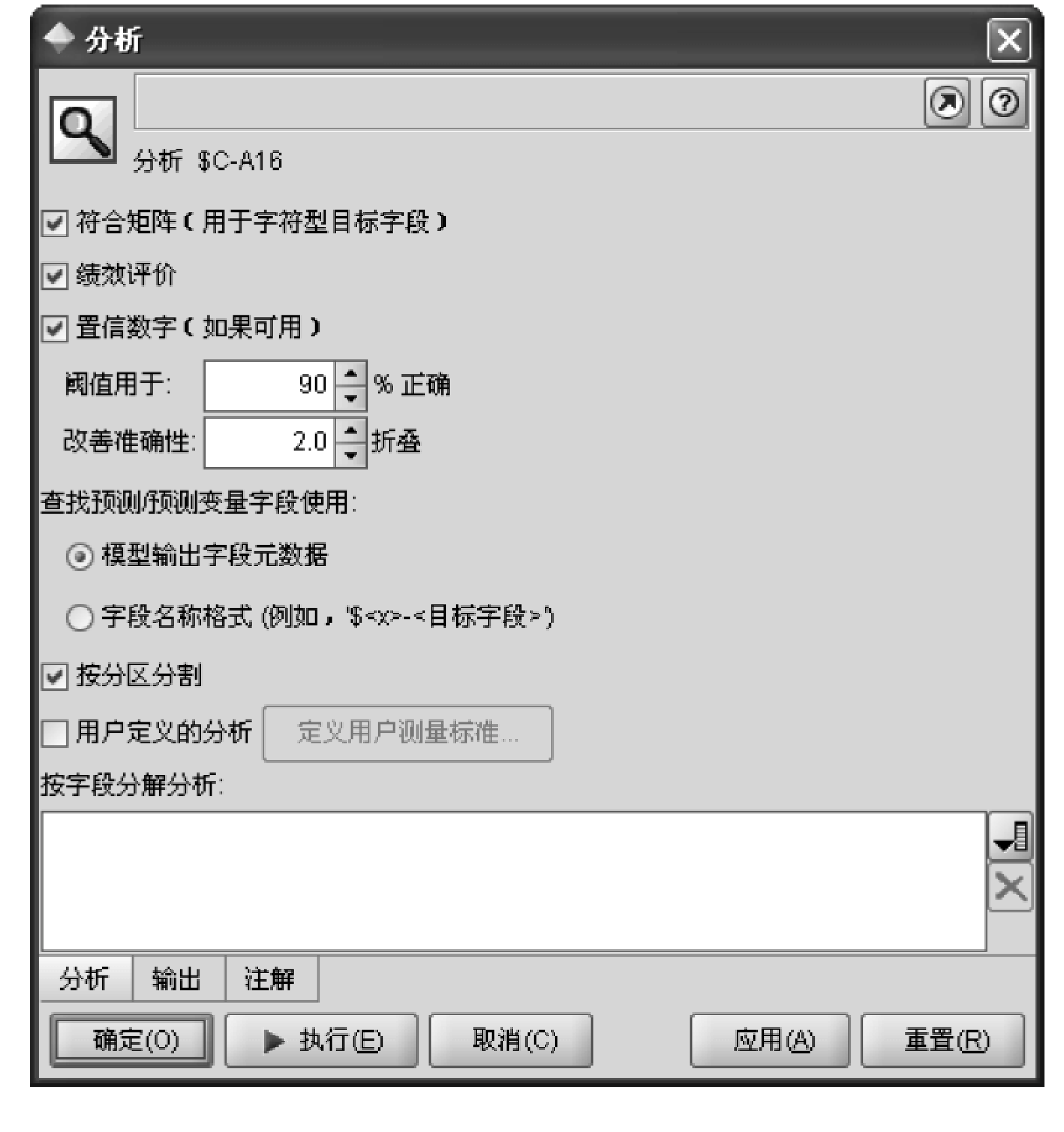


图 3.24　编辑“分析”节点的属性

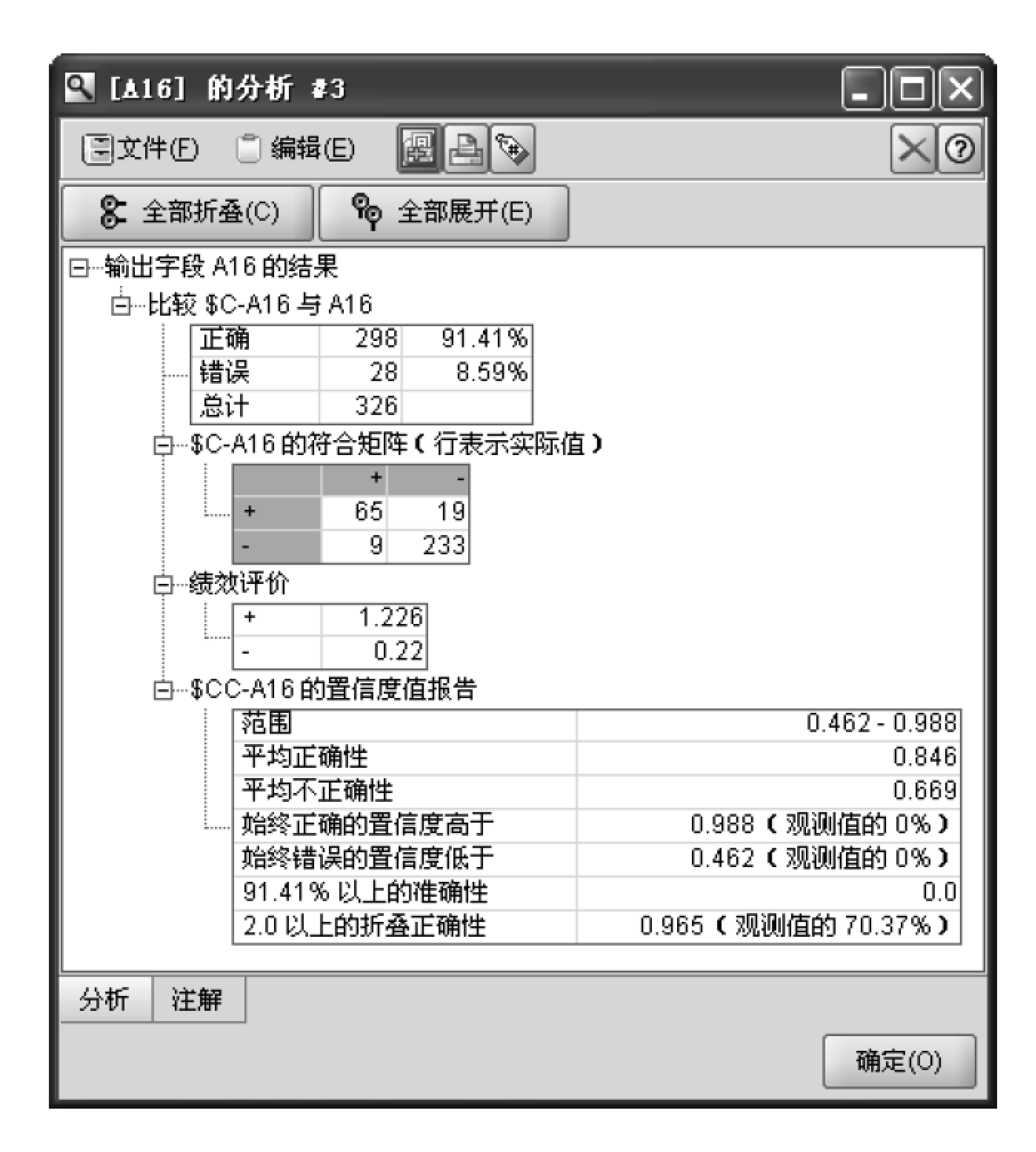


图 3.25　分析结果

从结果可以看出，有 91.41%的测试样本（298 个）的预测值和实际值相符，8.59%的测试样本（28 个）的预测值和实际值不符。在 28 个预测错误的样本中，实际值为“+”但被预测为“–”的样本有 19 个，实际值为“–”但被预测为“+”的样本有 9 个。对预测类别“+”和“–”的绩效评价统计量分别为 1.226 和 0.22。

置信度值报告的参数说明：

范围：对于在流中数据的记录，显示置信值的最小和最大的值。

平均正确性（Mean Correct）：对于被正确分类了的记录，显示平均置信度。

平均不正确性（Mean Incorrect）：对于没被正确分类了的记录，显示平均置信度。

始终正确的置信度高于（Always Correct Above）：显示一个置信度阈值，在此值以上的预测值总是正确的以及满足这个标准的个案的百分数。

始终错误的置信度低于（Always Incorrect Below）：显示一个置信度阈值，在此值以下的预测值总是错误的以及满足这个标准的个案的百分数。

X%以上的准确性（*X*% Accuracy Above）：显示精确度为 *X*%的置信水平。*X* 是对于在分析选项（Analysis Options）中被指定的阈值（Treshold for）的近似值。

X 以上的折叠正确性（Fold Correct Above）：显示了精确度是整个数据集的 *X* 倍的置信度。*X* 是在分析选项中的“改善准确性”值。

3.3 CART

分类和回归树（Classification and Regression Trees，CART，在 Clementine 中简写为 C&RT），由 Breiman 等在 1984 年出版的同名书中详细描述。CART 算法中的每一次分裂把数据分为两个子集，每个子集中的样本比被划分之前具有更好的一致性（属于同一类别的样本比例更高）。它是一个递归的过程，也就是说，这些子集还会被继续划分，这个过程不断重复，直到满足终止准则，然后通过修剪和评估，得到一棵最优的决策树。

CART 算法根据类标号属性的类型，可以构建分类树或者回归树模型。当类标号属性是连续型时，生成的模型是回归树；当类标号属性是离散型时，生成的模型是分类树。

给定样本集 *S*，用 CART 算法来构建决策树的过程大致可以分为 3 个步骤：生成最大树、树的修剪和子树评估。即先生成一棵充分生长的最大树，然后根据修剪算法对最大树进行修剪，生成由许多子树组成的子树序列，最后通过子树评估，从子树序列中选择一棵最优的子树作为最后的结果。

3.3.1 生成最大树

1. 标准问题集

在 ID3 算法中，数据集按照某个属性进行分裂时，可以产生多个分支（分支的数量取决于该属性有多少个不同的取值）。但 CART 不同，它的每次分裂只能产生两个分支，所以 CART 算法产生的决策树是一棵二叉树。

决策树算法中必不可少的步骤是为分裂确定一个分裂属性，即究竟按照哪一个属性

来把当前数据集划分为若干个子集，从而形成若干个“树枝”。同时，就某个给定的属性来说，由于属性的取值可能有很多个，所以按照这个属性来分裂数据集的方式也有很多种，属性的**标准问题集**就是所有候选分支方案的集合。在 CART 算法中，连续属性（数值属性）和离散属性（分类属性）的标准问题集的形式是不同的。

（1）连续属性的标准问题集

如果属性 A 是连续属性，那么 A 的标准问题集是由形如“Is $A\leqslant d$?”的测试条件构成的集合。首先将属性 A 的不同取值按大小顺序排列，然后依次计算每相邻两个数值的平均值，组成新的序列 A'，然后根据 A'产生标准问题集。

例如“年龄”这个连续属性的取值按大小顺序排列为{20，25，28，40，46，55，56，58，60，65，70}，那么中间值序列为{22.5，26.5，34，43，50.5，55.5，57，59，62.5，67.5}，于是属性 A 的标准问题集为：{Is $A\leqslant22.5$，Is $A\leqslant26.5$，Is $A\leqslant34$，Is $A\leqslant43$，Is $A\leqslant50.5$，Is $A\leqslant55.5$，Is $A\leqslant57$，Is $A\leqslant59$，Is $A\leqslant62.5$，Is $A\leqslant67.5$}。每一个标准问题都是一种划分方案，例如“Is $A\leqslant43$”可以根据年龄是否大于等于 43 岁把数据集划分为两个子集。

（2）离散属性的标准问题集

如果属性 A 是离散属性，那么 A 的标准问题集是由形如“Is $A\in s$?”的测试条件构成的集合。设属性 A 的不同取值组成的集合是 $S(A)$，那么由 $S(A)$的一些子集构成了集合 subset(A)，subset(A)中的每个元素满足以下两个条件：

- 如果 $s\in$subset(A)，那么 $s\neq S(A)$，且 $s\neq\varnothing$。
- 任意两个元素 $s_1\in$subset(A)和 $s_2\in$subset(A)，有 $s_1\cup s_2\neq S(A)$。

例如“方向”的不同取值组成的集合为{东,南,西,北}，那么 subset(方向)={{东},{东,南},{南},{东,西},{东,南,西},{南,西},{西}}。于是属性“方向”的标准问题集为{Is $A\in${东},Is $A\in${东,南},Is $A\in${南},Is $A\in${东,西},Is $A\in${东,南,西},Is $A\in${南,西},Is $A\in${西}}。

注意，之所以没有{西,北}，是因为已经有了{东,南}，而按照“Is $A\in${东,南}”和“Is $A\in${西,北}”来划分的结果是一样的，这就是上面第二个约束条件的含义所在。

2．杂度削减

前面提到过，决策树分裂的基本原则是，数据集被分裂为若干个子集后，要使每个子集中的数据尽可能的“纯”，也就是说子集中的记录要尽可能属于同一个类别。在 ID3 算法中，曾用“熵”来度量数据集随机性的程度。在 CART 中把这种随机性的程度称为“杂度”（impurity，也称为“不纯度”），并且用“吉尼”(gini)指标来衡量它。

设 t 是决策树上的某个节点，该节点的数据集为 S，由 s 个样本组成，其类标号属性具有 m 个不同的取值，即定义了 m 个不同的类 C_i（i=1,2,⋯,m）。设属于类 C_i 的样本的个数为 s_i。那么这个节点的吉尼指标这样来计算：

$$\text{gini}(t)=1-\sum_{i=1}^{m}(p(C_i\mid t))^2$$，其中，$p(C_i\mid t)=\frac{s_i}{s}$ 表示类 C_i 在数据集 S 中的概率。

因此，如果数据集 S 中的样本非常平均地分布在各个类别中，数据集的杂度就最高，此时的 gini 指标有最大值（$1-1/k$），其中，k 表示类标号有多少个不同的取值。如果全部样本都属于同一类别，此时的 gini 指标则等于 0。

由于 CART 算法生成的是一棵二叉树，所以对于节点 t 来说，分裂后将产生两个子节点 t_L 和 t_R，设这两个子节点的杂度分别为 gini(t_L)和 gini(t_R)，那么，在此次分裂过程中的杂度削减为：

$$\Phi(S,t)=\text{gini}(t)-p_L\text{gini}(t_L)-p_R\text{gini}(t_R)$$

其中，p_L 是 t 中的样本被划分到 t_L 的概率，p_R 是 t 中的样本被划分到 t_R 的概率。

根据每一个属性的各种标准问题，计算在各种分支方案下的杂度削减值，最后选择杂度削减量最大的属性作为分裂属性，相应的标准问题就是最佳的分支方案。

对于分支后产生的子节点，采用上面同样的方法，继续对这些节点进行划分，直到满足了某个停止准则才停止分裂，最后生成一棵完全生长的二叉树，这棵树称为最大树 T_{max}。

下面是一个计算杂度削减的例子：

数据集 S 在节点 1 以属性 $A15$ 根据“≤500”和“>500”分裂为两个子集，对应于节点 2 和节点 12，如图 3.26 所示。

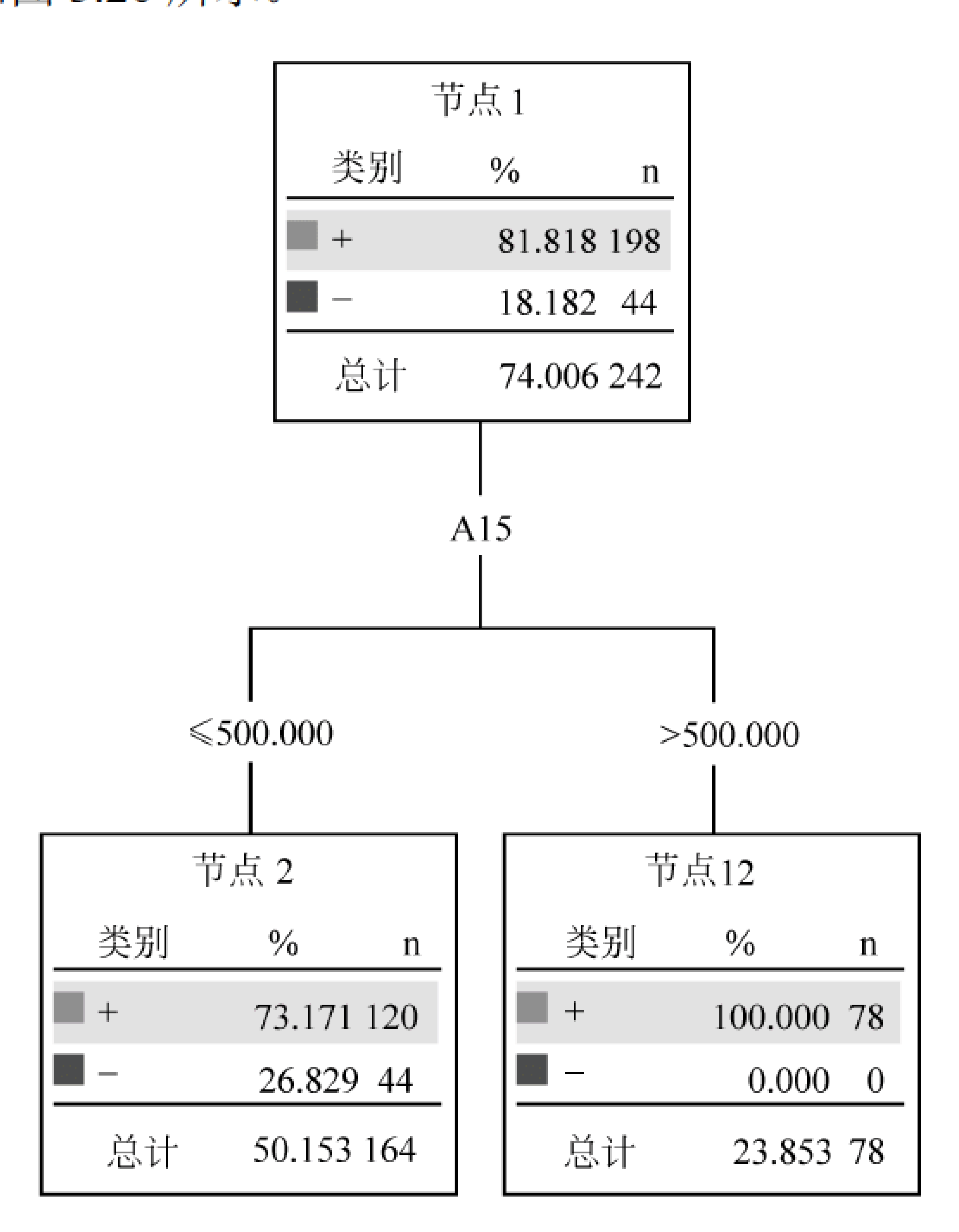


图 3.26 计算杂度削减

节点 1 的杂度为：gini（节点 1）$=1-(0.818^2+0.182^2)=0.298$

节点 2 的杂度为：gini（节点 2）$=1-(0.732^2+0.268^2)=0.392$

节点 12 的杂度为：gini（节点 12）$=1-(1^2+0^2)=0$

所以，杂度削减为：

$$\Phi(S,\text{节点 }1)=\text{gini}(\text{节点 }1)-p_{\text{节点 }2}\text{gini}(\text{节点 }2)-p_{\text{节点 }12}\text{gini}(\text{节点 }12)$$

$$=0.298-\frac{164}{242}\times0.392-\frac{78}{242}\times0=0.032$$

3. 停止准则

停止准则用来控制何时停止节点的继续分裂。只有当每个叶子节点都促发了至少一个停止准则时，树的生长过程才完全停止下来。对于一个节点来说，以下任何一个规则被满足，该节点都将不再分裂：

- ❑ 这个节点是“纯”的，即这个节点的所有样本都属于同一类别。
- ❑ 对于每一个属性（不包括类标号属性），节点中的所有样本都有相同的值。
- ❑ 这个节点所在的深度已经达到“最大树深度”（如果定义有）。
- ❑ 这个节点的样本数量小于“父分支中的最小记录数”（如果定义有）。
- ❑ 这个节点分裂后产生的子节点中包含的样本数量小于预定义的“子分支中的最小记录数”（如果定义有）。
- ❑ 分裂产生的杂度削减小于预定义的“最小杂度削减”（如果定义有）。

4. 其他杂度度量

在 Clementine 中用 CART 建立决策树模型时，对数据集杂度的度量指标有 3 种，根据数据集的类标号属性的类型有所不同。如果类标号属性是离散型，即用 CART 来构建分类树，可以选择“gini 指标”或者“Twoing（两分）指标”来度量杂度；如果类标号属性是连续型，即用 CART 来构建回归树，则用“最小平方误差”（Least-Squared Deviation，LSD）。

对于 Twoing 指标的度量方法，首先根据类标号将数据集划分为两个超类，然后在两个超类的基础上去寻找能实施最佳分裂的某个属性。两个超类 C_1 和 C_2 的定义如下：

$$C_1 = \{j : p(j|t_L) \geqslant p(j|t_R)\}, \quad C_2 = C - C_1$$

其中，C 是类标号属性的全部取值组成的集合，j 表示类标号的某个取值，$p(j|t_L)$ 表示类标号等于 j 的样本被划分到子集 t_L 的概率，$p(j|t_R)$ 表示类标号等于 j 的样本被划分到子集 t_R 的概率。

设节点 t 以属性 s 被划分为两个子集 t_L 和 t_R，那么 Twoing 指标为：

$$\Phi(s,t) = p_L p_R \left[\sum_j | p(j|t_L) - p(j|t_R) | \right]^2$$

取使得 $\Phi(s,t)$ 为最大值的属性 s 为分裂属性。

3.3.2 树的修剪

通过上面的步骤，可以得到一棵根据训练样本集充分生长的决策树——最大树 T_{max}。所谓“充分生长”，就是说在分裂生长过程中并没有给予其他的限制，只是根据训练样本集来不断分裂出子节点。这样的充分生长的决策树模型往往会有很多的叶子节点。由于理解和解释具有大量叶子节点的决策树是一个复杂的过程，所以这样的决策树具有很高的“**复杂度**”，复杂度可以用叶子节点的数量来衡量。我们要尝试着对树进行修剪，使叶子节点尽可能少。但是如果修剪不当或者过度修剪，虽然复杂度降低了，但却会大大降低决策树模型的分类精确度，导致一定的“**误分类损失**”，付出一定的“**代价**”。

由于想让误分类损失和复杂度这两个指标都最小，所以用一个指标来综合评估误分类损失和复杂度，即**代价复杂度**：

$$R_\alpha(T)=R(T)+\alpha|\tilde{T}|$$

其中，$R_\alpha(T)$ 是树 T 的代价复杂度，$R(T)$是树 T 的误分类损失，$|\tilde{T}|$是树 T 的叶子节点数量。α 是复杂度参数，表示每增加一个叶子节点所提高的代价复杂度。当 $\alpha=0$ 时，表示不考虑节点数量对代价复杂度的影响，在构建最大树的时候正是如此。显然，代价复杂度越小越好。

假设决策树上有一个节点 t，包含两个子节点 t_1 和 t_2，如图 3.27 所示。

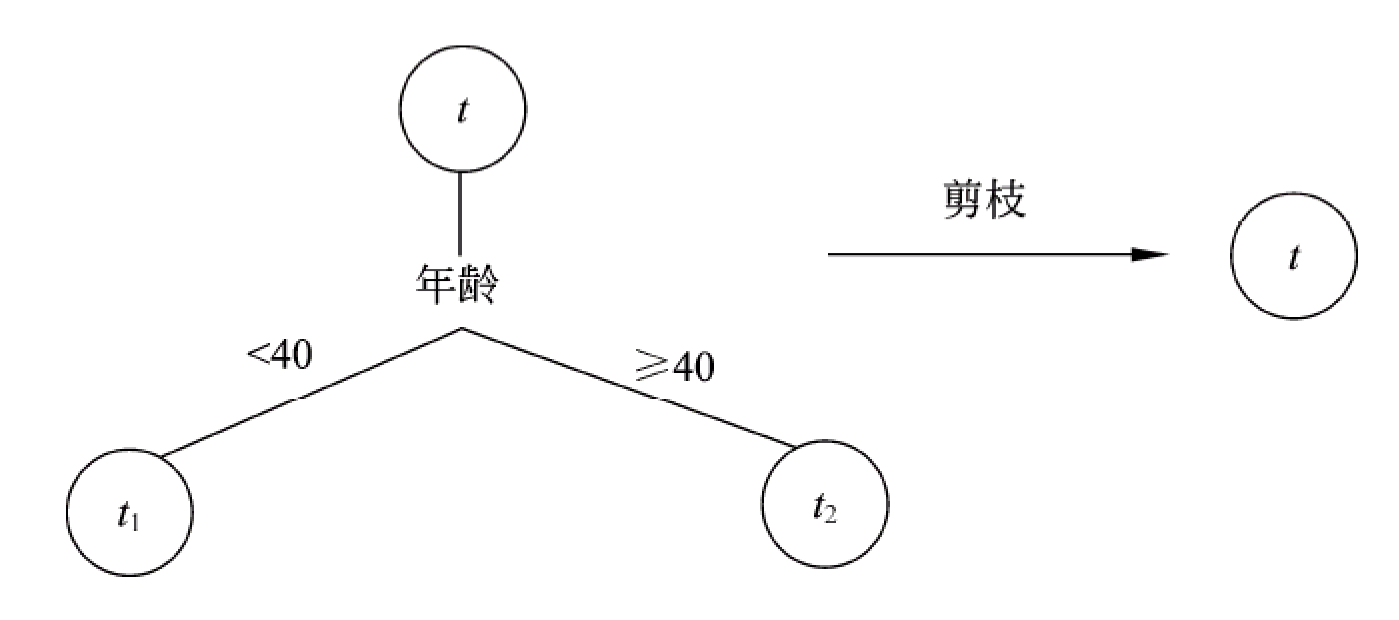


图 3.27　决策树的一次修剪

把图 3.27 中左边的这棵树记做 T_t，表示由节点 t 生长出来的树，那么图中所示的这次修剪，实际上就是把 T_t的分支删除，使节点 t 成为一个叶子节点。

那么修剪之前的代价复杂度，也就是 T_t的代价复杂度为：

$$R_\alpha(T_t)=R(T_t)+\alpha|\tilde{T}_t|$$

修剪之后的代价复杂度，也就是节点 t 的代价复杂度（节点可以理解成“只有一个节点的树”）为：

$$R_\alpha(\{t\})=R(t)+\alpha$$

显然，如果 $R_\alpha(\{t\})>R_\alpha(T_t)$，或者说，节点 t 被分裂成树 T_t之后，使得代价复杂度减小了，T_t的存在才是有价值的，就应该被保留下来。而树在完全生长过程中（此时 $\alpha=0$）的每一次分裂也确实保证了这一点。换句话说，当最大树 T_{max} 要进行修剪的时候，其实每个节点的分支都不应该被修剪。那么现在的问题是，选择树上的哪一个节点进行修剪呢？

虽然 T_{max} 上的节点都满足 $R_\alpha(\{t\})>R_\alpha(T_t)$，但前提条件是 $\alpha=0$。就 T_{max} 上的某个节点 t 而言，随着 α 的增大，$R_\alpha(\{t\})$ 和 $R_\alpha(T_t)$ 都会线性增大，其中后者增大更快。最后会有一个阈值 α'，使得当 $\alpha\geqslant\alpha'$时有 $R_\alpha(\{t\})\leqslant R_\alpha(T_t)$。这表示当 α 超过 α' 后，如果修剪掉节点 t 下面的分支，可以降低代价复杂度，这时就应该执行修剪。

根据不等式 $R_\alpha(\{t\})\geqslant R_\alpha(T_t)$，可以推出这个阈值为：

$$\alpha'=\frac{R(t)-R(T_t)}{|\tilde{T}_t|-1}$$

对树上的每一个节点，计算它们的这个阈值。假设其中阈值最小的那个节点为$\overline{t}$，它的阈值为：$g(\overline{t})=\min_{t\in T} g(t)$。

所以，随着 α 由 0 不断增大，节点$\overline{t}$ 是第一个使得 $R_\alpha(\{t\})=R_\alpha(T_t)$ 成立的节点，称

节点$\bar{t}$与树 T 具有“**最弱连接**”，它就是修剪的对象。

有了这些背景知识，下面可以归纳一下树的修剪过程：

- 令 $\alpha=0$，从 $T_1=T(0)$开始，这里的 $T(0)$就是最大树 $T_{\max}$。
- 逐渐增大 α，直到某个节点使得 $R_\alpha(\{t\})=R_\alpha(T_t)$ 成立，将它的分支删除，得到 T_2。
- 重复上一步骤，直到被修剪到只有一个根节点，从而得到一个树的序列 $T_1,T_2,\cdots,T_k$。

显然，$T_1,T_2,\cdots,T_k$ 都是 $T_{\max}$ 的子树，而且节点数量依次减少。下一步，要对这些子树进行评估，从中选择一棵最优树作为最后的决策树模型。

3.3.3 子树评估

子树评估的目的就是从上面得到的子树序列 $T_1,T_2,\cdots,T_k$ 中，按照某种评估方法找到一棵最优的树。简单地说，要找到一棵分类准确性最好、同时节点数量尽量少的树。

常用的评估方法是计算每一棵子树的误分类损失：

$$R(T)=\frac{1}{N}\sum_{i\neq j}c(i\mid j)N_{ij}$$

其中，$R(T)$是树 T 的误分类损失，N 是数据集中的样本个数，$c(i\mid j)$ 是将类 j 误分类为类 i 的损失，N_{ij} 表示将类 j 误分类为类 i 的样本个数。

当样本数量比较少的时候，可以直接用训练样本来计算这些树的误分类损失，即“重替代评估”。另外，也可以采用交叉验证评估。

当样本数量比较多时，可以用独立于训练样本的测试样本来计算这些树的误分类损失。

最后，$R(T)$最小的树，就被选择出来作为最优树。

然而，对于同一棵树，不同的数据集计算出来的误分类损失往往就不同。说到底，这里计算出来的误分类损失其实都是一个估计值。或许存在这样一棵树，虽然它的误分类损失比最小误分类损失大一点点，但节点数量却少一些，而它才是最优的树。

因此，Breiman 等提出了所谓的 1SE 规则（One Standard Error）。既然误分类损失是一个估计量，那么可以计算出它的标准误（Standard Error），记做 $\mathrm{SE}(R(T))$，然后先把那些误分类损失落在区间$[\min\limits_k R(T_k),\min\limits_k R(T_k)+\mathrm{SE}(R(T))]$中的树挑选出来（$\min\limits_k R(T_k)$是 k 棵子树的误分类损失中的最小值），最后从这些树中选择那个节点数量最少的树，成为最后的决策树模型。也就是说，1SE 规则扩大了最优树的选择范围，而不是直接选择误分类损失最小的树。

在 CART 的应用中，默认使用一倍标准误，用户也可以自行指定多倍标准误。综合这些情况，子树评估就是从子树序列中挑选出满足以下公式的规模最小的树：

$R(T_{\mathrm{opt}})\leqslant\min\limits_k R(T_k)+m\times\mathrm{SE}(R(T))$，$T_{\mathrm{opt}}$ 就是最优树。

其中，m 默认为 1。如果 $m=0$，表示直接根据误分类损失来选择最优树。当 $m>1$ 时，m 越大，则最优树的规模越小（和误分类损失最小的那棵树相比）。

这种选择机制的好处在于，它可以避免选择那些过度拟合或者拟合不足的树。

3.3.4 在 Clementine 中应用 CART

这是一个利用 CART 进行市场分析的案例。其研究目的是根据调查数据构建一个分类树模型，来预测哪些客户有意向订购一个电视新闻服务。该案例在 Clementine 的自带文档中有所描述。

数据集存放在文件...\clementine12.0\Demos\NewsChan.sav 中，是 Clementine 的自带文件。该数据集的属性包括：EDUCATE（受教育年限）、GENDER（性别）、AGE（年龄）、TVDAY（每天看电视的时间长度，即小时数）、ORGS（组织编号）、CHILDS（孩子的个数）、INC（收入）、NEWSCHAN（类标号属性，表示是否愿意订购有线新闻服务）。

完整的数据流如图 3.28 所示。

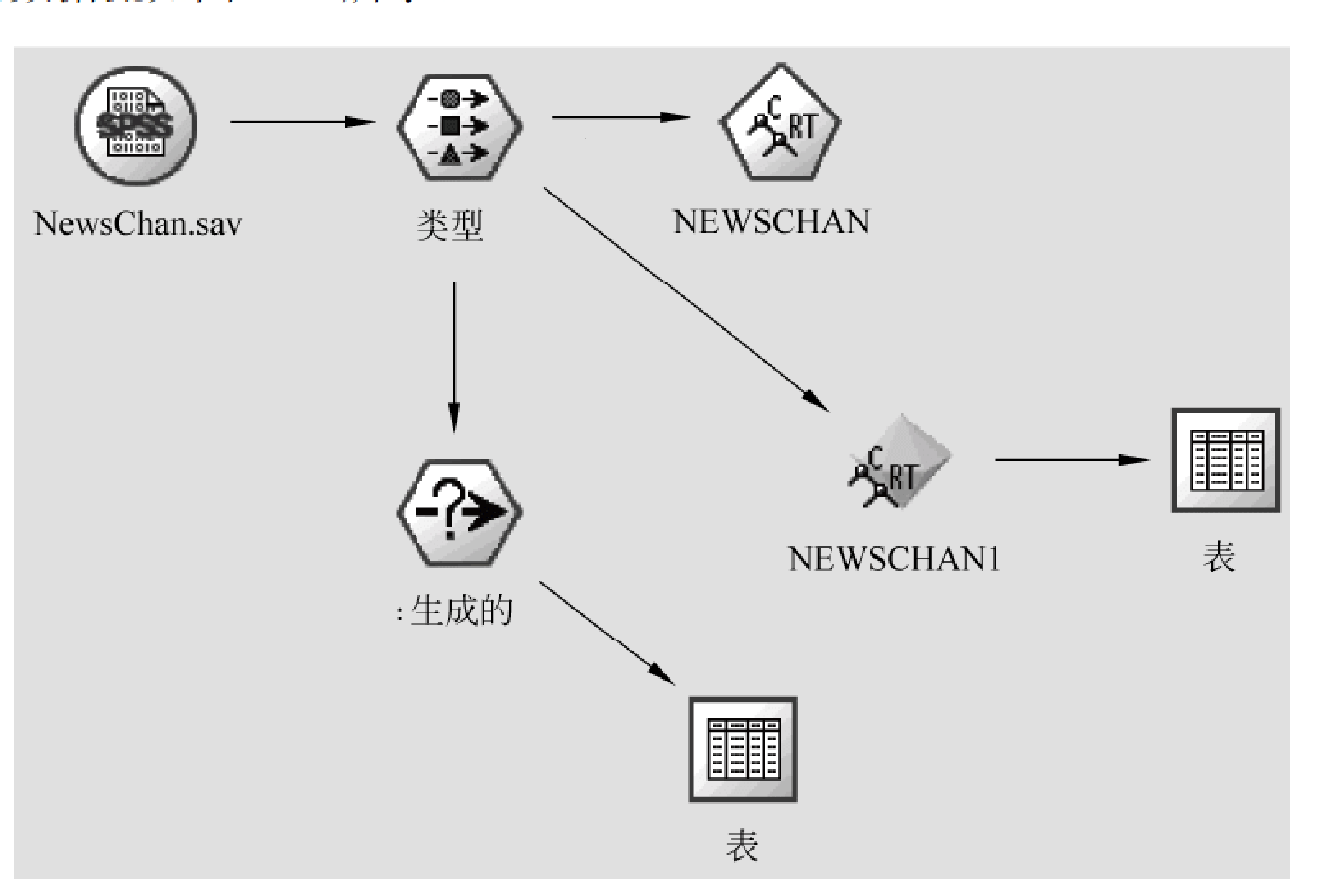


图 3.28 市场分析案例数据流

下面着重阐述具体的建模和分析过程。

1. 构建决策树

（1）在数据流区域添加“SPSS 文件”节点，并编辑其属性，导入 Demos 文件夹中的 NewsChan.sav 文件。如果想浏览数据，可以在其后添加“表”节点并执行。

（2）在数据流区域添加“类型”节点，并编辑其属性，单击“读取值”按钮，然后将 NEWSCHAN 字段的“类型”设置为“标志”型，并把“方向”设置为“输出”（如图 3.29 所示）。

（3）添加 C&RT 节点，并建立从“类型”节点到 C&RT 节点的连接。编辑 C&RT 节点。在“模型”标签下，选中“启动交互会话”单选按钮（如图 3.30 所示），这样在执行该节点时会启动 “交互树”窗口，允许用户在生成模型之前进行一些编辑操作。



图 3.29　编辑“类型”节点



图 3.30　设置“启动交互会话”

在“专家”标签下，选中“专家”模式，可由用户自己来设置一些参数（如图 3.31 所示）。

最大代用项（Maxmum Surrogates）：代理是处理缺失值的一种方法。对于树中的每一次拆分，分类回归树识别与拆分字段最相似的输入字段，这些字段是该拆分字段的代理。如果必须对某一记录分类，而该记录拆分字段有缺失值，则该记录拆分字段的代理字段值可用于拆分。提高该项设置值使对缺失值的处理更加灵活，但是也可能增加内存使用大小和训练次数。这里设置为 5。

最小杂质改变（Minimum Change In Impurity）：指定在树中进行新的拆分所需的最小杂度改变量。如果某一分支的最优拆分引起的杂度改变量低于指定值，则不会进行拆分。这里设置为 0.003。增大这个数值会阻止那些杂度改变很小的分裂，从而趋向于生成一棵简单的树。

“分类目标的杂质测量”选择 Gini 单选按钮，即用 gini 指标来度量杂度。

选中“修剪树”和“使用标准误规则”复选框，“乘数”默认为 1.0，即使用 1 倍标准误。

单击“正在停止...”按钮，可以设置停止标准（如图 3.32 所示）。当节点触发任一准则时，该节点则停止分裂。



图 3.31　设置专家模式下的参数

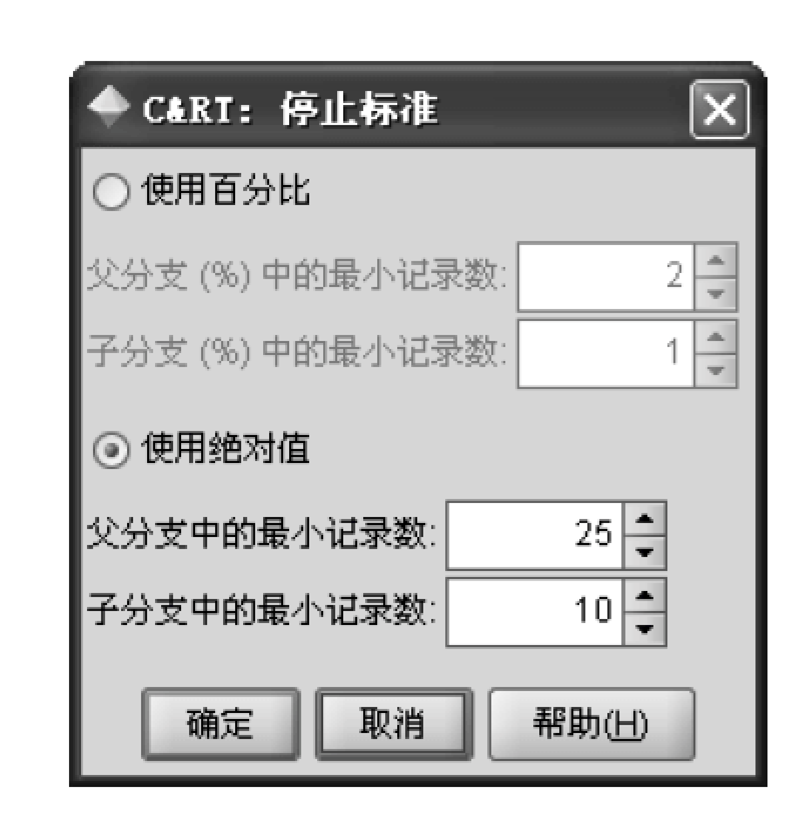


图 3.32　设置停止标准

父分支中的最小记录数（Minimum Records In Parent Branch）避免在被拆分节点（父节点）记录数小于指定值时对该节点的拆分。

子分支中的最小记录数（Minimum Records In Child Branch）避免在拆分节点所生成每一分支记录数均小于指定值时对该节点的拆分。

设置最小记录数可以采用百分比方式或者绝对值方式。

（4）执行该节点，将打开交互树窗口（如图 3.33 所示），允许用户进行一些编辑操作。



图 3.33　交互树窗口

可以看到现在只生成了一个根节点。统计数据表示，训练数据集包含 442 个样本。由于树还没有开始分裂，全部样本均属于该节点。在全部样本中，有 215 个样本表示愿意订购服务，响应比例为 48.643%。

在“树”菜单中选择“生长树并修剪”，即可生成一棵决策树（如图 3.34 所示），树深为 5 层，叶子节点有 5 个。

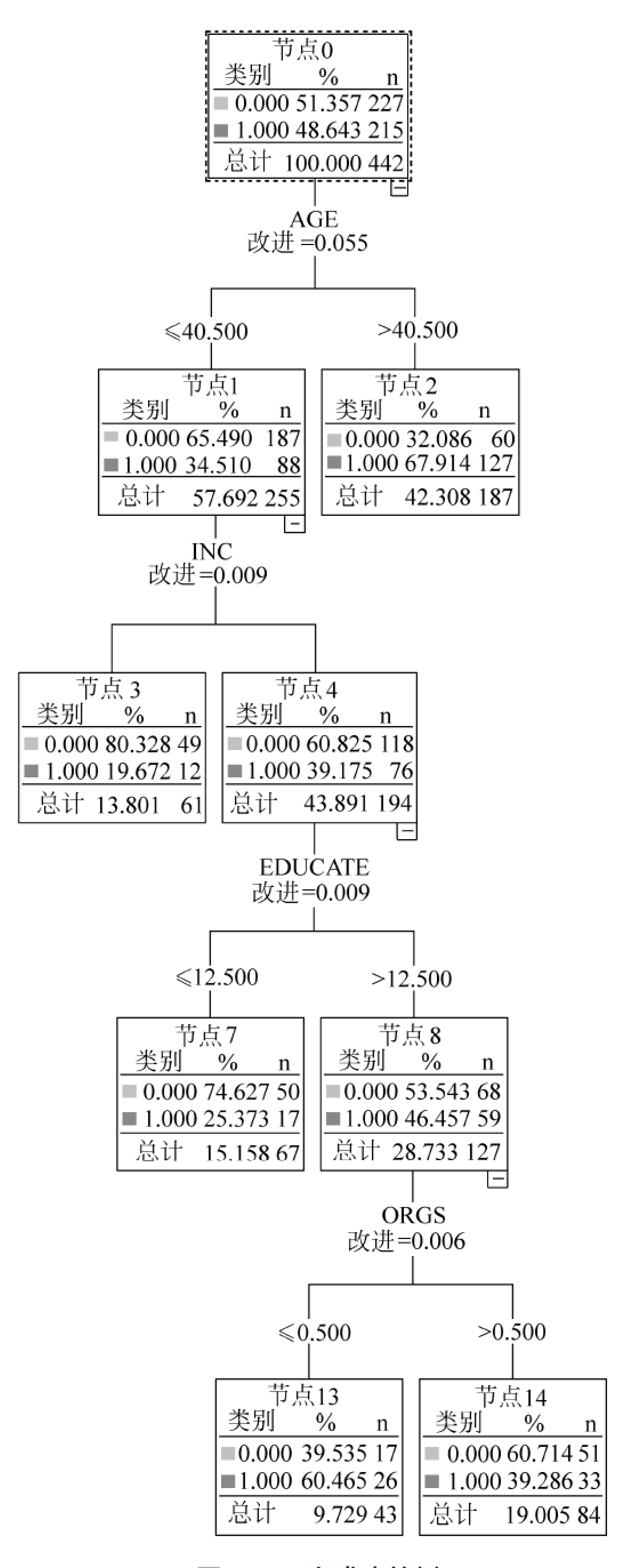


图 3.34 生成决策树

决策树的显示大小（缩放）、方向（由上至下、由左至右）皆可调节。

2．对决策树的分析

从上面的决策树可以看出，样本集首先按照年龄分裂，然后按照收入分裂。从节点 2 可以看出，如果把年龄限制在 40.5 岁以上，响应率将提高到 67.914%。

虽然年龄小于 40.5 的人群响应率要低得多，但是如果再加以一些条件的限制，同样还可以提高响应率，比如节点 13 的响应率就达到了 60.465%。

下面来分析一下这棵树的收益表。收益比例可以告诉人们，对于某个类别的样本，它们在某个分支节点中的比例和在根节点中的比例相比，究竟有多大的提高。

（1）单击交互树窗口的“增益”标签，就可以得到收益表，选择“目标类别”为 1.0（如图 3.35 所示）。



节点	节点: n	节点 (%)	收益: n	收益 (%)	响应 (%)	指数 (%)
2	187.00	42.31	127.00	59.07	67.91	139.62
13	43.00	9.73	26.00	12.09	60.47	124.31
14	84.00	19.00	33.00	15.35	39.29	80.76
7	67.00	15.16	17.00	7.91	25.37	52.16
3	61.00	13.80	12.00	5.58	19.67	40.44

图 3.35　收益表

表中按照降序列出了 5 个叶子节点的相关参数值。下面以节点 13 为例，来说明表中的这些参数的含义：

- 节点:n（43.00）：节点 13 中一共有 43 个样本。
- 节点(%)（9.73）：节点 13 中的样本量占总样本量的 9.73%。
- 收益:n（26.00）：节点 13 中有 26 个样本类别为“1.0”。
- 收益(%)（12.09）：由于在节点 13 中有 26 个样本类别为“1.0”，而在根节点中有 215 个样本类别为“1.0”，这两个数的比值为：26/215=12.09%。
- 响应(%)（60.47）：节点 13 中有 60.47%的样本，其类别为“1.0”，26/43=60.47%。
- 指数(%)（124.31）：由于在节点 13 中有 60.47%的样本类别为“1.0”，而在根节点中有 48.643%的样本类别为“1.0”，这两个数的比值为：60.47%/48.643%=124.31%。这个参数表示，对一些属性进行限制之后得到的样本集和初始样本集相比，响应率提高了 124.31%。

从表中可以看出，节点 2 和节点 13 具有最高的指标值。它们的指标值大于 100%，

这就表示从这些节点中随机抽取的用户和从根节点中随机抽取的用户相比，会有更高的几率接受新闻订购服务。比如节点 2，该节点中的用户接受订购服务的几率比根节点中的样本高 1.39 倍。

（2）在交互树窗口“增益”标签中单击“分位数”按钮，并从下拉列表中选择“十分位数”；然后单击“图表”按钮，并从旁边的下拉列表中选择“提升”，就可以看到指数的提升图（如图 3.36 所示）。

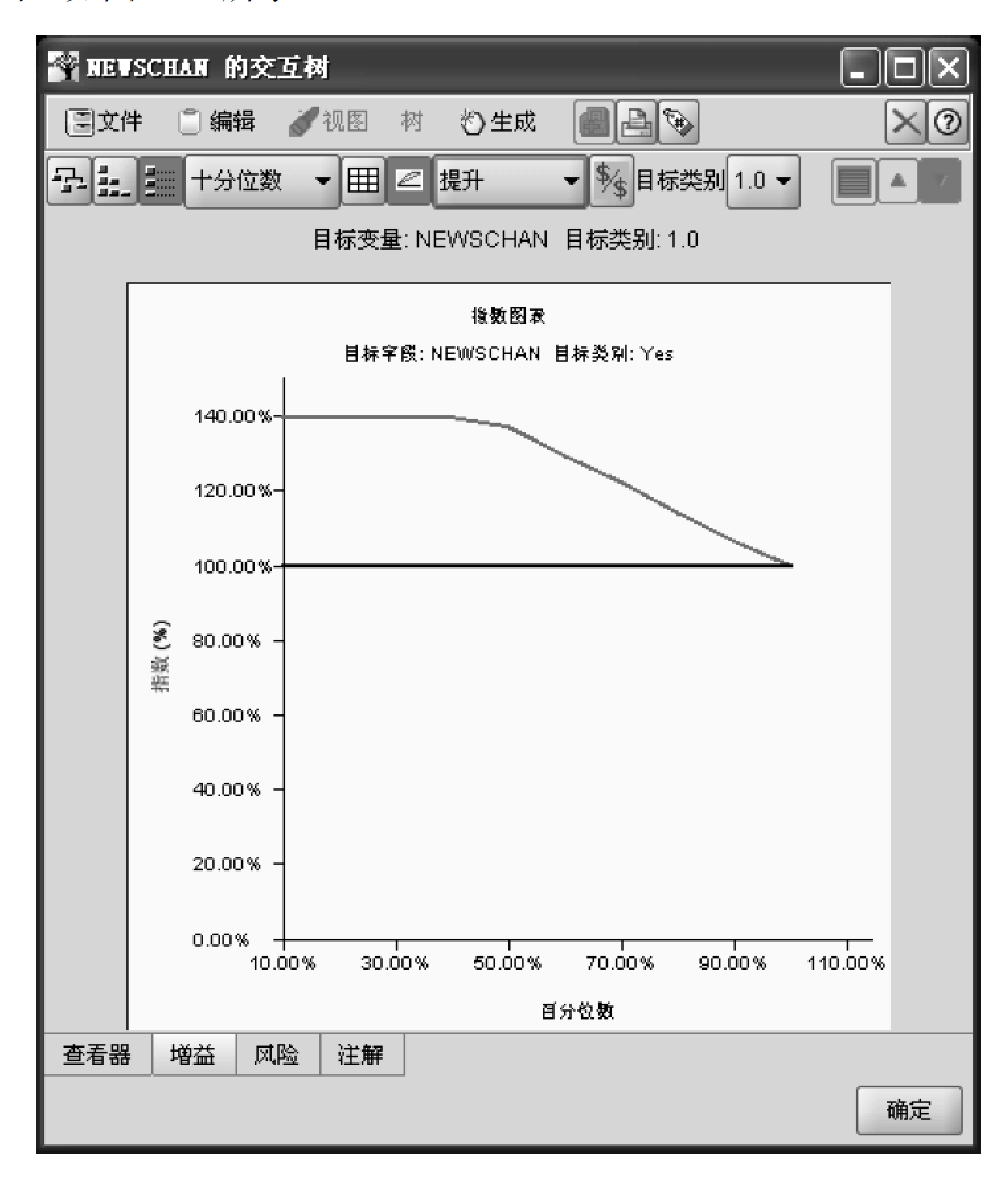


图 3.36　指数提升图

Lift（提升指数）是评估一个预测模型是否有效的一个度量，这个比值由运用和不运用这个模型所得来的结果计算而来。

在指数提升图中，横坐标是抽取样本数的百分比，纵坐标是指数值。从图中可以看出，可以找到大约 50%的响应概率较高的样本（由节点 2 和节点 13 组成）来使效率提高约 140%。

3．生成“选择”节点

既然节点 2 和节点 13 的样本是“优质”样本，可以根据它们的属性条件来自动生成一个“选择”节点。用这个节点，可以直接从测试样本集或者预测样本集中选择出响应

概率较高的样本。操作步骤如下：

（1）在交互树窗口的“查看器”标签下，先单击选中节点 2，然后按住 Ctrl 键不放，单击节点 13。这样，两个节点都被选中。

（2）单击“生成”菜单下的“选择节点”命令，即可在数据流区域生成一个“选择”节点。双击该节点，可以看到该节点中自动生成的选择条件，如图 3.37 所示。

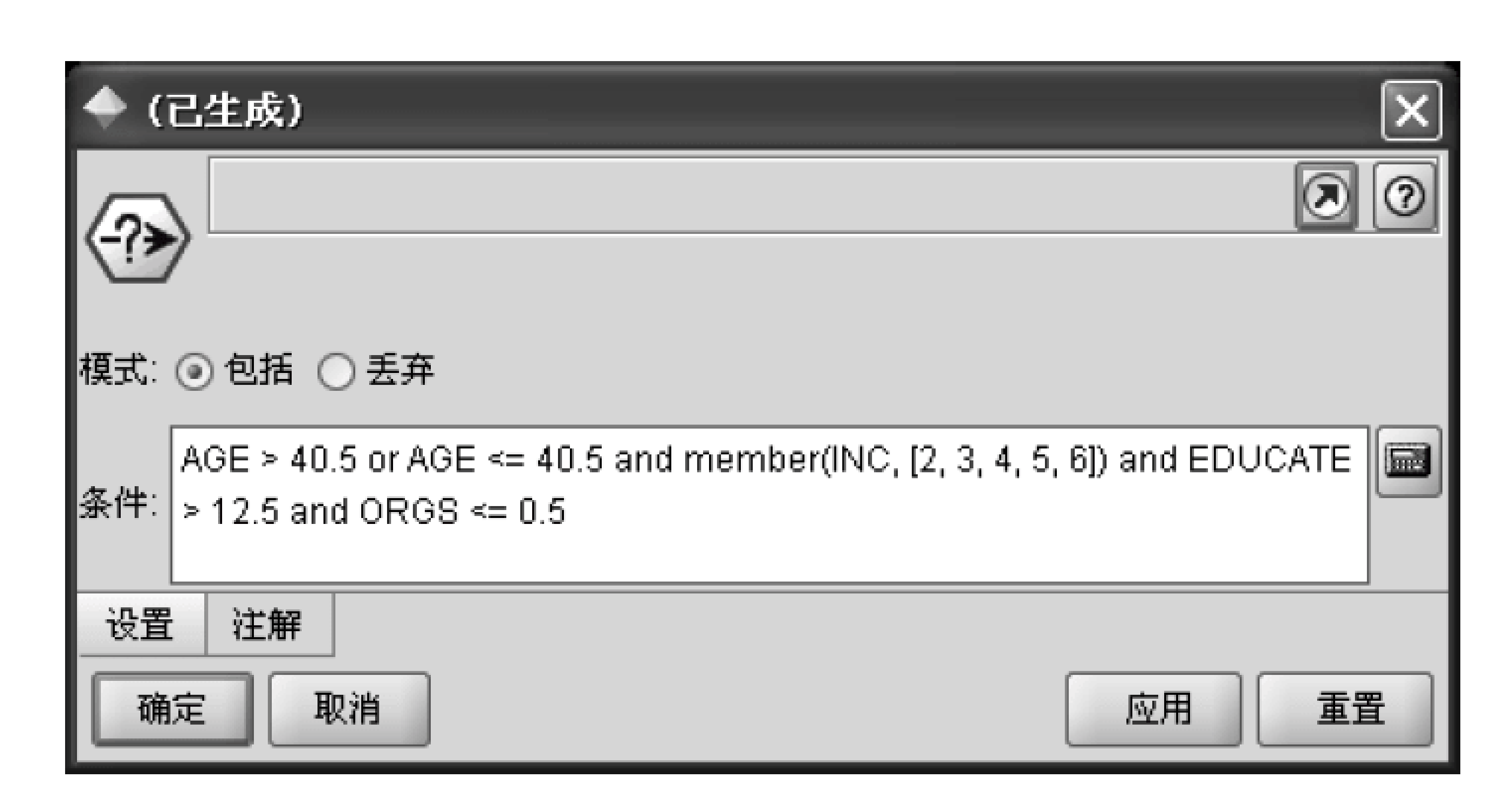


图 3.37　自动生成的选择条件

如果此时建立由“类型”节点到这个“选择”节点的连接，再在“选择”节点后面连接一个表节点（如图 3.28 所示）并执行表节点，即可看到从训练样本集中选择出来的“优质”样本。

4．生成模型

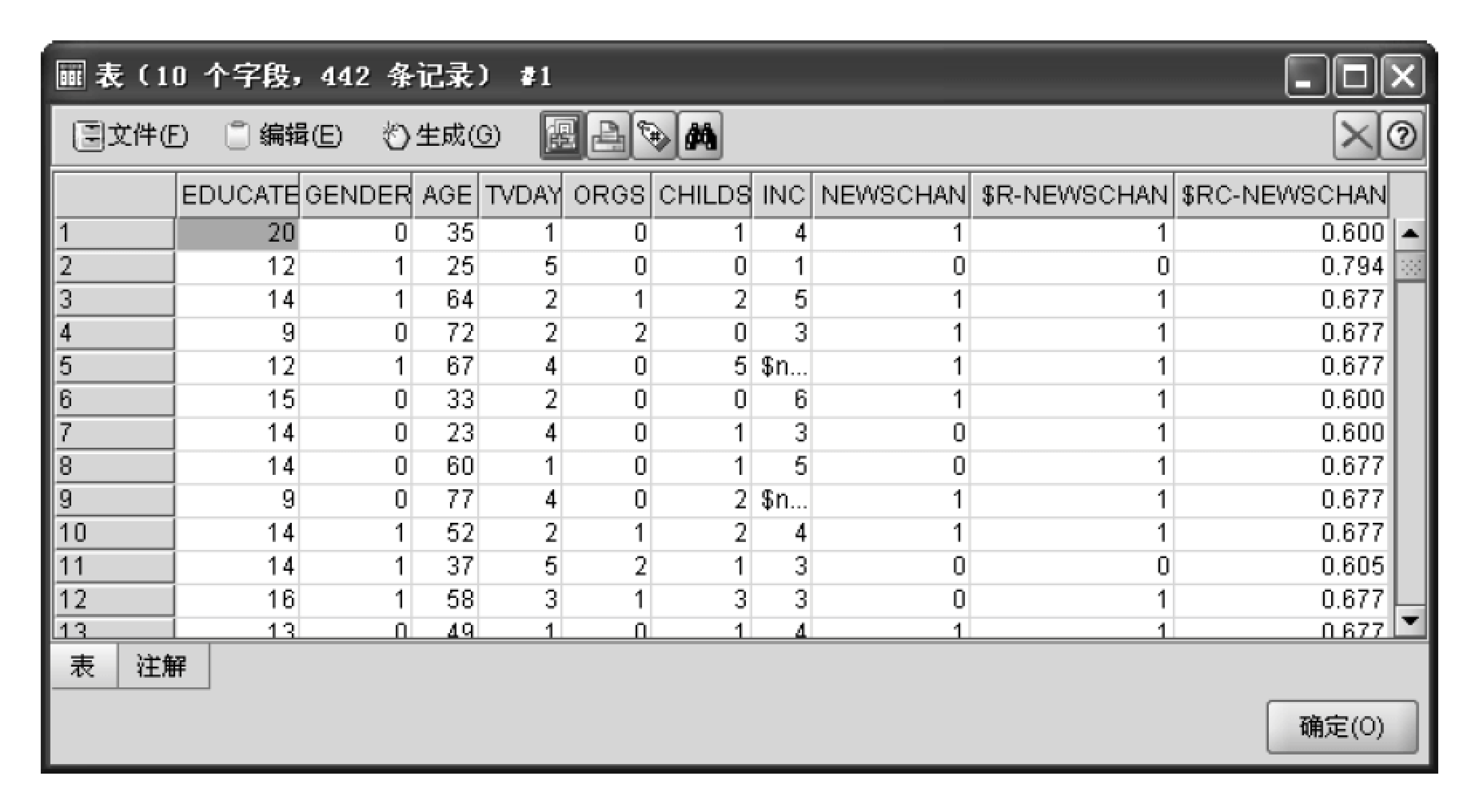
（1）单击交互树窗口中“生成”菜单下的“生成模型…”命令，打开“生成新的模型”对话框，如图 3.38 所示。



图 3.38　“生成新的模型”对话框

“模型名称”可以选择“自动”，也可以选择“自定义”并输入名称。创建节点位置如果选择“工作区”，将在数据流区域产生模型节点；如果选择“GM 选项板”，则在管理器窗口的“模型”标签下产生；也可以选择“两者”，则在两个位置同时产生。

（2）建立由“类型”节点到新产生的 NEWSCHAN1 之间的连接（见图 3.28），即把训练数据集带入生成的决策树模型。再在 NEWSCHAN1 之后连接一个表节点并执行之，即可看到训练数据集中每个样本的分类结果和置信度，如图 3.39 所示。



	EDUCATE	GENDER	AGE	TVDAY	ORGS	CHILDS	INC	NEWSCHAN	$R-NEWSCHAN	$RC-NEWSCHAN
1	20	0	35	1	0	1	4	1	1	0.600
2	12	1	25	5	0	0	1	0	0	0.794
3	14	1	64	2	1	2	5	1	1	0.677
4	9	0	72	2	2	0	3	1	1	0.677
5	12	1	67	4	0	5	$n...	1	1	0.677
6	15	0	33	2	0	0	6	1	1	0.600
7	14	0	23	4	0	1	3	0	1	0.600
8	14	0	60	1	0	1	5	0	1	0.677
9	9	0	77	4	0	2	$n...	1	1	0.677
10	14	1	52	2	1	2	4	1	1	0.677
11	14	1	37	5	2	1	3	0	0	0.605
12	16	1	58	3	1	3	3	0	1	0.677
13	13	0	49	1	0	1	4	1	1	0.677

图 3.39　对训练样本的分类结果

其中，NEWSCHAN 列是实际值，$R-NEWSCHAN 列是预测值，$RC- NEWSCHAN 列是置信度。

第 4 章　聚 类 分 析

4.1 聚类分析概述

4.1.1 聚类分析的概念

1. 无指导学习

聚类与第 3 章讨论的分类有相似之处，都是将数据进行分组，但两者又有本质的区别。分类中的组（类别）是事先已经定义好的，但聚类中的组（在聚类分析中称为“簇”）不是预先定义的，而是根据实际数据的特征按照数据之间的相似性来定义的。

在分类问题中，训练样本的分类属性（类标号）的值是已知的，而在聚类问题中，需要在训练样本中确定这个分类属性值。采用聚类分析技术，可以把无标识的数据样本自动划分为不同的类，并且可以不受人的先验知识的约束和干扰，从而获取属于数据集中原本存在的信息。

所以说，聚类是一种无指导学习（无监督学习），分类则是一种有指导学习（有监督学习）。无指导学习是从样本的特征向量出发研究通过某种算法将特征相似的样本聚集在一起，从而达到区分具有不同特征样本的目的；有指导学习的最大特点是具有先验知识（类标号），而无监督聚类学习并不具有这种先验知识。无指导学习主要用于聚类，有指导学习主要用于回归与分类问题。

2. 聚类分析的定义

聚类分析（Cluster Analysis）又称群分析，是根据“物以类聚”的道理，对样品或指标进行分类的一种多元统计分析方法。通过聚类分析，可以在没有任何模式可供参考或依循，即在没有先验知识的情况下，将大量数据样本按各自的特性来进行合理的分类。

聚类，就是把整个数据集分成不同的“簇”，并且要使簇与簇之间的区别尽可能的大，而簇内的数据的差异尽可能的小。簇是数据样本的集合，聚类分析使得每个簇内部的样本之间的相关性比与其他簇中样本之间的相关性更紧密，即簇内的任意两个样本之间具有较高的相似度，而属于不同簇的两个样本间具有较高的相异度。相异度可以根据描述样本的属性值来计算，样本间的“距离”是最常采用的度量标准。

在开始聚类之前，用户并不知道要把数据集分成几个簇，也不知道划分的具体标准，在聚类分析时数据集的特征是未知的，聚类算法的任务正是要发现这些特征，并把具有相同特征的数据样本聚在一起。

聚类分析的数学定义如下：

给定数据集 V：$\{s_1,s_2,\cdots,s_n\}$，其中 s_i（i=1,2,⋯,n）为数据样本。根据数据样本之间的相似程度将数据集分成 k 个簇：$C_1,C_2,\cdots,C_k$，且满足：

$$C_i \subseteq V；\ C_i \cap C_j = \varnothing；\ \bigcup_{i=1}^{k} C_i = V，\ 1 \leqslant i \leqslant k，\ 1 \leqslant j \leqslant k，\ i \neq j。$$

3．聚类分析的特征

聚类分析是根据事物本身的特性研究个体的一种方法，目的在于将相似的事物归类。它的原则是同一类中的个体有较大的相似性，不同类的个体差异性很大。这种方法有以下 3 个特征：

（1）适用于没有先验知识的分类。如果没有这些事先的经验或一些国际标准、国内标准、行业标准，分类便会显得随意和主观。这时只要设定比较完善的分类变量，就可以通过聚类分析法得到较为科学合理的类别。

（2）可以处理多个变量决定的分类。例如，要根据消费者购买量的大小进行分类比较容易，但如果在进行数据挖掘时，要求根据消费者的购买量、家庭收入、家庭支出、年龄等多个指标进行分类通常比较复杂，而聚类分析法可以解决这类问题。

（3）聚类分析法是一种探索性分析方法，能够分析事物的内在特点和规律，并根据相似性原则对事物进行分组，是数据挖掘中常用的一种技术。

4.1.2 聚类分析的基本方法

聚类分析的研究已经有很多年的历史，研究成果主要集中在基于距离和基于相似度的方法上，也产生了大量的聚类算法。算法的选择取决于数据的类型、聚类的目的和应用。如果聚类分析被用做描述或探查的工具，可以对同样的数据尝试多种聚类算法，以发现数据中隐藏的规律。

大体上，主要的聚类算法可以划分为如下几类：

1．划分方法（Partitioning Method）

给定一个包含 n 个样本的数据集，划分方法将构建数据集的 k 个划分，每个划分表示一个簇，同时满足如下的要求：

（1）每个组至少包含一个样本。

（2）每个样本必须属于且只属于一个簇。

注意在某些模糊划分技术中第二个要求可以放宽。

给定要构建的划分的数目 k，划分方法首先创建一个初始划分。然后采用一种迭代的重定位技术，尝试通过对象在划分间移动来改进划分。一个好的划分的一般准则是：在同一个类中的对象之间尽可能“接近”或相关，而不同类中的对象之间尽可能“远离”或不同。还有许多其他划分质量的评判准则。

典型的划分方法包括：K-Means、K-Medoids、CLARA（Clustering Large Application）、CLARANS（Clustering Large Application based upon Randomized Search）、FCM 等。

2. 层次的方法（Hierarchical Method）

层次的方法对给定数据集进行层次的分解。根据层次的分解过程不同，层次的方法可以分为凝聚的和分裂的。凝聚的方法，也称为自底向上的方法，一开始将每个样本作为单独的一个组，然后相继地合并相近的对象或组，直到所有的组合并为一个（层次的最上层），或者达到一个终止条件。分裂的方法，也称为自顶向下的方法，一开始将所有的样本置于一个簇中，在迭代的每一步中，一个簇被分裂为更小的簇，直到最终每个样本在单独的一个簇中，或者达到一个终止条件。

层次的方法的缺陷在于，一旦一个步骤（合并或分裂）完成，它就不能被撤销。这个严格规定是有用的，由于不用担心组合数目的不同选择，计算代价会较小。但是，该方法的一个主要问题是它不能更正错误的决定。有两种方法可以改进层次聚类的结果，其一，在每层划分中，仔细分析对象间的“连接”，例如 CURE（Clustering Using Reprisentatives）和 Chameleon 中的做法；其二，综合层次凝聚和迭代的重定位方法，首先用自底向上的层次算法，然后用迭代的重定位来改进结果。例如在 BIRCH（Balanced Iterative Reducing and Clustering using Hierarchies）中的方法。

3. 基于密度的方法（Density-based Method）

绝大多数划分方法基于样本之间的距离进行聚类。这样的方法只能发现球状的簇，而在发现任意形状的簇上遇到了困难。随之提出了基于密度的另一类聚类方法，其主要思想是：只要临近区域的密度（对象或数据点的数目）超过某个阈值，就继续聚类。也就是说，对给定类中的每个数据点，在一个给定范围的区域中必须至少包含某个数目的点。这样的方法可以用来过滤“噪声”孤立点数据，发现任意形状的簇。

DBSCAN（Densit-based Spatial Clustering of Application with Noise）是一个有代表性的基于密度的方法，它根据一个密度阈值来控制簇的增长。OPTICS（Ordering Points To Identify the Clustering Structure）是另一个基于密度的方法，它为自动的和交互的聚类分析计算一个聚类顺序。

4. 基于网格的方法（Grid-based Method）

基于网格的方法把样本空间量化为有限数目的单元，形成了一个网格结构。所有的聚类操作都在这个网格结构（即量化的空间）上进行。这种方法的主要优点是它的处理速度很快，其处理时间独立于数据样本的数目，只与量化空间中每一维的单元数目有关。

STING（Statistical Information Grid）是基于网格方法的一个典型例子。 CLIQUE（Clustering In Quest）和 WaveCluster 这两种算法既是基于网格的，又是基于密度的。

5. 基于模型的方法（Model-based Method）

基于模型的方法为每个簇假定了一个模型，寻找数据对给定模型的最佳拟合。一个基于模型的算法可能通过构建反映数据点空间分布的密度函数来定位聚类。它也基于标准的统计数字自动决定聚类的数目，考虑“噪声”数据或孤立点，从而产生健壮的聚类方法。典型的基于模型的方法包括：

统计方法 COBWEB：是一个常用的且简单的增量式概念聚类方法。它的输入对象是采用符号量（属性-值）对来加以描述的。采用分类树的形式来创建一个层次聚类。

CLASSIT 是 COBWEB 的另一个版本。它可以对连续取值属性进行增量式聚类。它为每个节点中的每个属性保存相应的连续正态分布（均值与方差）；并利用一个改进的分类能力描述方法，即不像 COBWEB 那样计算离散属性（取值）和而是对连续属性求积分。但是 CLASSIT 方法也存在与 COBWEB 类似的问题。因此它们都不适合对大数据库进行聚类处理。

传统的聚类算法已经比较成功地解决了低维数据的聚类问题。但是由于实际应用中数据的复杂性，在处理许多问题时，现有的算法经常失效，特别是对于高维数据和大型数据的情况。因为传统聚类方法在高维数据集中进行聚类时，主要遇到两个问题。首先，高维数据集中存在大量无关的属性使得在所有维中存在簇的可能性几乎为零。另外，高维空间中数据较低维空间中数据分布要稀疏，其中数据间距离几乎相等是普遍现象，而传统聚类方法是基于距离进行聚类的，因此在高维空间中无法基于距离来构建簇。

高维聚类分析已成为聚类分析的一个重要研究方向。同时高维数据聚类也是聚类技术的难点。随着技术的进步使得数据收集变得越来越容易，导致数据库规模越来越大、复杂性越来越高，如各种类型的贸易交易数据、Web 文档、基因表达数据等，它们的维度（属性）通常可以达到成百上千维，甚至更高。但是，受“维度效应”的影响，许多在低维数据空间表现良好的聚类方法运用在高维空间上往往无法获得好的聚类效果。高维数据聚类分析是聚类分析中一个非常活跃的领域，同时它也是一个具有挑战性的工作。

4.2 K-Means 算法

K-Means 算法，也称 k-平均算法，是一种常用的基于划分的聚类方法，用来根据样本属性值之间的相似度来对样本进行分组。其基本思路是，把数据集划分为 k 个簇，每个簇内部的样本都非常相似，但不同簇的样本则非常相异。K-Means 是一种迭代算法，初始的 k 个簇被随机地定义之后，这些簇将被不断地更新，并在更新中被优化，当无法再进一步优化（或者达到一定的迭代次数）时算法才停止，然后生成模型。

在 K-Means 算法中，每个簇有一个中心，称为“质心”，k 个簇就相应地有 k 个质心。一个样本究竟被划分到哪个簇，就看它和哪个质心的“相异度”最小。在 K-Means 算法中，衡量相异度的指标是“距离”（distance）。所以也可以这么说，一个样本究竟被划分到哪个簇，就看它和哪个质心的“距离”最小。这里的距离，则是由样本的每一个属性的取值来共同参与决定的。

4.2.1 数据预处理

为了进行聚类，需要不断地计算样本之间的相异度，即距离。两个样本之间的距离又是由多个距离分量（有多少个属性参与聚类分析，就有多少个距离分量）来计算的，每个距离分量就是两个样本同一属性的取值差异。为了计算这些距离，要首先对数据集中的数据进行预处理。

在一个数据集中，往往既有连续属性，也有离散属性。两种属性的预处理方式则完全不同。

1. 连续属性的预处理

在一个数据集中，往往会有多个连续属性，例如“年龄”、“家庭拥有汽车数量”等。对于“年龄”来说，最大值可以达到 80 甚至更大，但对于“家庭拥有汽车数量”而言，通常不会超过 3~4 辆。为了平抑属性间的这种差异，把连续属性的值都统一地转换为一个属于区间[0,1]的值，转换的公式是：$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$，其中，$x_{\min}$ 和 $x_{\max}$ 分别是属性 X 的全部取值中的最小值和最大值，x_i 是某个样本的属性 X 的取值，x_i' 就是转换后的属性值。

例如样本 A 和样本 B 的属性 X 的转换如表 4-1 所示。

表 4-1　连续属性值的转换示例

样本	属性 X 的取值	属性 X 的最小值	属性 X 的最大值	转换后的取值
A	28	10	80	0.257
B	54	10	80	0.629

那么这两个样本在属性 X 上的差异 d 可以这样来计算：

$$d = (x_A - x_B)^2 = (0.257 - 0.629)^2 = 0.138$$

显然，差异 d 的最大值为 1（此时两个取值分别为属性值中的最小值和最大值），差异 d 的最小值为 0（此时两个取值相等）。

2. 离散属性的预处理

离散属性的取值往往不是数值，例如“收入水平”的取值包括“高”、“中”、“低”。因此，为了便于计算属性值的差异，必须先将离散属性连续化。

一种常用的方法是对离散属性的全部取值进行二进制编码，用多个连续属性来表示一个离散属性，例如对“收入水平”的转换过程显示在表 4-2 中。

表 4-2　离散属性值的转换示例

样本	收入水平	S_1	S_2	S_3
A	中	0	1	0
B	高	1	0	0
C	低	0	0	1

这样，“收入水平”可以由 3 个连续属性 S_1、S_2、S_3 来表示，这 3 个属性称为原属性的“派生属性”。

那么样本 A 和样本 B 在属性“收入水平”上的差异该如何计算呢？如果采用上面计算连续属性差异的方法，则：

$$d = (S_{1A} - S_{1B})^2 + (S_{2A} - S_{2B})^2 + (S_{3A} - S_{3B})^2 = (0-1)^2 + (1-0)^2 + (0-0)^2 = 2$$

计算结果大于 1，意味着这种计算方法将使得在计算样本之间的距离时，离散属性

的影响（权重）会比连续属性更大。为了平抑这种差异，需要引入一个调节因子 $\eta=\sqrt{0.5}=0.707$。这样，样本 A 和样本 B 在属性“收入水平”上的差异为：

$$d=(\eta S_{1A}-\eta S_{1B})^2+(\eta S_{2A}-\eta S_{2B})^2+(\eta S_{3A}-\eta S_{3B})^2$$
$$=(0-\sqrt{0.5})^2+(\sqrt{0.5}-0)^2+(0-0)^2=1$$

显然，当样本 A 和样本 B 的离散属性值相等时，差异 d 为 0，否则为 1。

4.2.2 K-Means 算法流程

K-Means 算法是一个迭代计算“质心”并根据样本与质心的距离把各样本指派到各个簇的过程。主要过程如下：

第 1 步：确定初始质心。生成 k 个质心，k 由用户指定。

第 2 步：指派样本。计算每一个样本到各质心的距离，把样本指派给距离最小的簇。

第 3 步：更新质心。根据每个簇当前所拥有的所有样本，重新计算每个簇的质心。

第 4 步：检查是否满足下列两个停止准则之一（如果满足，则算法停止。否则重复第 2 步和第 3 步）：

- 在第 3 步结束时，更新后的质心和更新前的质心之间的区别小于预定义的差异容忍度。
- 迭代次数已经超过了预定义的次数。

一个簇是由它的质心来代表的。质心是一个向量（由属性的值构成）。向量的值是由指派给该簇的那些样本的属性值的平均值来确定的。

下面对算法流程中的各个步骤进行阐述。

1. 确定初始质心

首先由用户指定模型中簇的数量 k。然后根据如下算法生成 k 个质心：

① 选取第一个样本，作为第一个质心。

② 对每一个样本，计算它与质心的欧几里得距离或者距离的平方。

③ 选择欧几里得距离最大的那个样本，作为又一个质心。

④ 重复②、③，直到 k 个质心都被确定下来。

给定质心向量 $C(c_1,c_2,\cdots,c_Q)$和一个样本向量 $X(x_1,x_2,\cdots,x_Q)$，其中 Q 是数据集中属性的数量，x_q 是样本的第 q 个属性的值，$q=1,2,\cdots,Q$。那么样本与质心的欧几里得距离的计算公式如下：

$$d=\sqrt{\sum_{q=1}^{Q}(x_q-c_q)^2}$$

最初的 k 个质心生成后，算法开始进行迭代指派过程。

2. 指派样本

在算法的每一次迭代中，每一个样本被指派给离自己最近的那个质心所代表的簇。距离是由欧几里得距离的平方来表示的，所以，样本 i 到质心 j 的距离为：

$$d_{ij} = \| X_i - C_j \|^2 = \sum_{q=1}^{Q} (x_{qi} - c_{qj})^2$$

其中，X_i是样本 i 的属性值组成的向量，C_j是簇 j 的质心向量，Q 是属性的数量，x_{qi} 是第 i 个样本的第 q 个属性的值，c_{qj} 是簇 j 的质心的第 q 个属性的值。

对每一个样本，计算它和每一个质心的距离。哪个簇的质心和该记录最近，就把该记录指派给那个簇。在这个过程中，一个样本可能会被从一个簇中转移到另一个簇中。

在所有的记录都被指派后，开始更新每个簇的质心。

3．更新质心

由于在指派样本的过程中，一个簇中的一些样本可能被转移到其他簇中（因为这些样本距离其他簇的质心更近），也可能会有一些其他簇中的样本被转移到这个簇中，因此需要重新计算每个簇的质心。

设指派样本之后，第 j 个簇中的样本数量为 m_j，那么重新计算这个簇的质心所得到的向量为：

$$\overline{X_j} = (x_{1j}, x_{2j}, \cdots, x_{Qj})$$

其中向量的第 q（q=1,2,⋯,Q）个分量 x_{qj} 为：

$$x_{qj} = \frac{\sum_{i=1}^{m_j} x_{qi}(j)}{m_j}$$

其中，$x_{qi}(j)$ 是簇 j 中的样本 i 的第 q 个属性的值。

4．停止准则

（1）最大迭代次数

“最大迭代次数”控制着算法不断地寻找更稳定的簇。算法将不断地重复“指派样本-更新质心”的循环，直到达到“最大迭代次数”。当这个极限达到之后，算法就终止更新簇，产生最终的模型。

（2）差异容忍度

“差异容忍度”提供另外一种控制算法是否终止的方式。

在每一次迭代结束后，计算每个簇更新前和更新后的质心间的距离。例如在第 t 次迭代结束后，第 j 个簇在更新前和更新后的质心间的距离为：$\| C_j(t) - C_j(t-1) \|$，其中 $C_j(t)$ 是第 t 次迭代时第 j 个簇的质心向量，$C_j(t-1)$ 是前一次迭代时第 j 个簇的质心向量。这样，k 个簇就产生了 k 个结果。选出其中的最大值 $\max_j \| C_j(t) - C_j(t-1) \|$，如果这个最大值小于预先定义的差异容忍度，则算法终止。否则继续迭代。

4.2.3 在 Clementine 中应用 K-Means

本节展示了一个在 Clementine 中用 K-Means 算法对某超市中的 15 种液体饮料进行聚

类分析的案例。目的是根据饮料中的 5 种微量元素的含量（毫克/升）来对这 15 种饮料进行分组。数据集存放在“饮料.xls”文件中，包含 15 个样本，由 5 个属性（*A*、*B*、*C*、*D*、*E*）组成（如表 4-3 所示）。

表 4-3　训练样本集

编号	*A*	*B*	*C*	*D*	*E*
1	12.1	40.86	448.7	0.012	1.01
2	18.4	42.61	467.3	0.008	1.64
3	32.3	12.86	325.61	0.004	2.22
4	27.2	9.18	369.8	0.005	1.72
5	8.9	57.67	556.55	0.018	1.01
6	16.2	36.18	425.78	0.003	1.594
7	25.3	10.86	348.7	0.002	2.01
8	5	47.79	540.13	0.017	0.77
9	17.2	38.2	424.4	0.001	1.14
10	11.4	34.23	405.6	0.008	1.02
11	25.5	17.35	346	0	1.78
12	17.2	33.67	443.2	0.001	1.414
13	10.2	40.01	516.7	0.012	0.95
14	5.4	40.12	530.8	0.014	0.63
15	20.8	33.02	445.8	0.004	1.618

完整的数据流如图 4.1 所示。

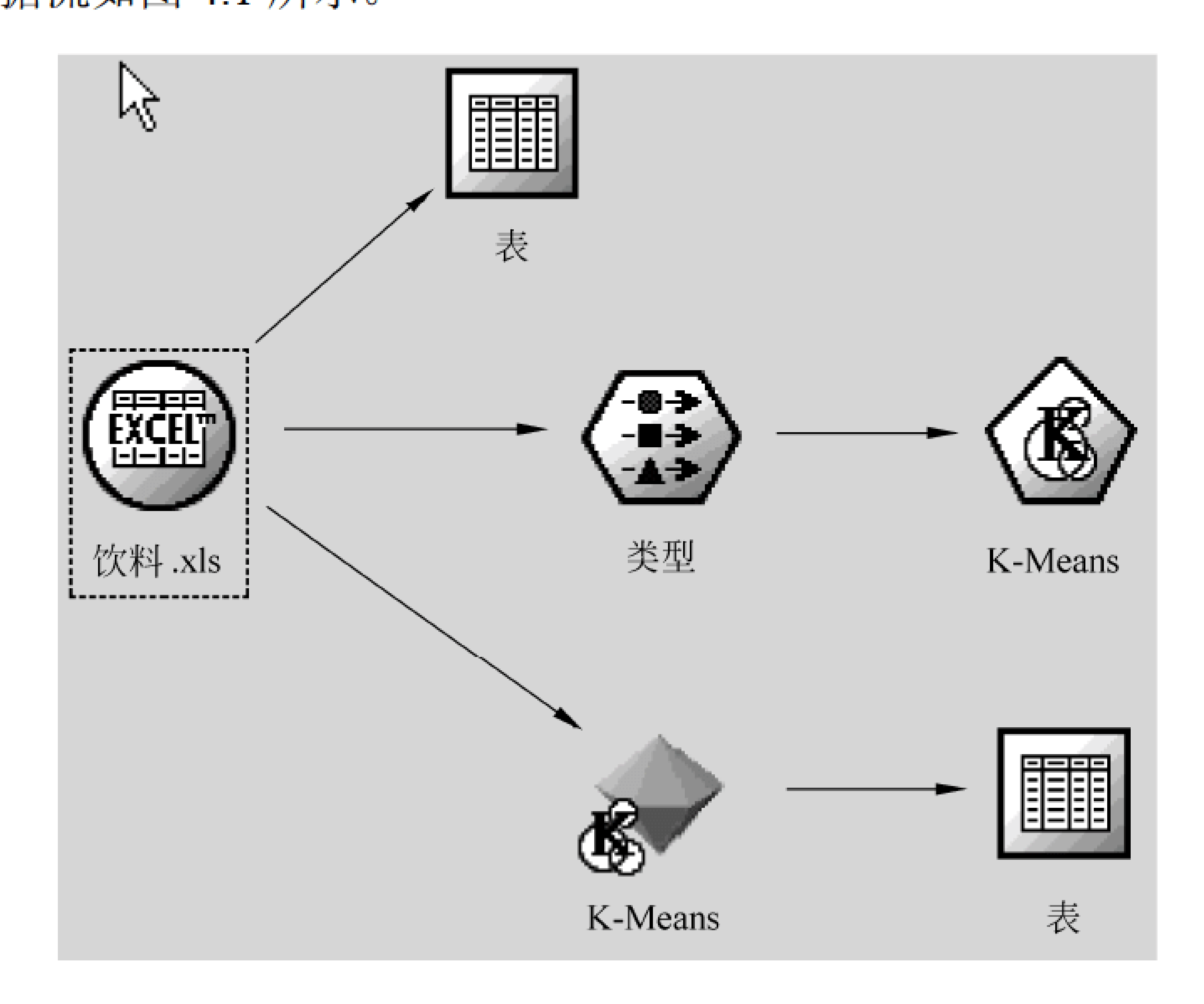


图 4.1　饮料 K-Means 聚类案例的数据流

1．设置数据来源

首先，在“数据源”选项板下选择 Excel 节点添加到数据流区域，并编辑其属性：在“饮料.xls”对话框的“数据”标签下，设置“导入文件”为“文件路径\饮料.xls”；

另外，数据集的第一列是“编号”，这只是样本的编号，显然该属性不应参与聚类分析，因此，在“饮料.xls”对话框的“过滤”标签下，将属性“编号”过滤掉（单击该属性右侧的箭头），如图 4.2 所示，然后单击“确定”按钮。如果想浏览数据，可以在“饮料.xls”节点后添加“表”节点并执行。

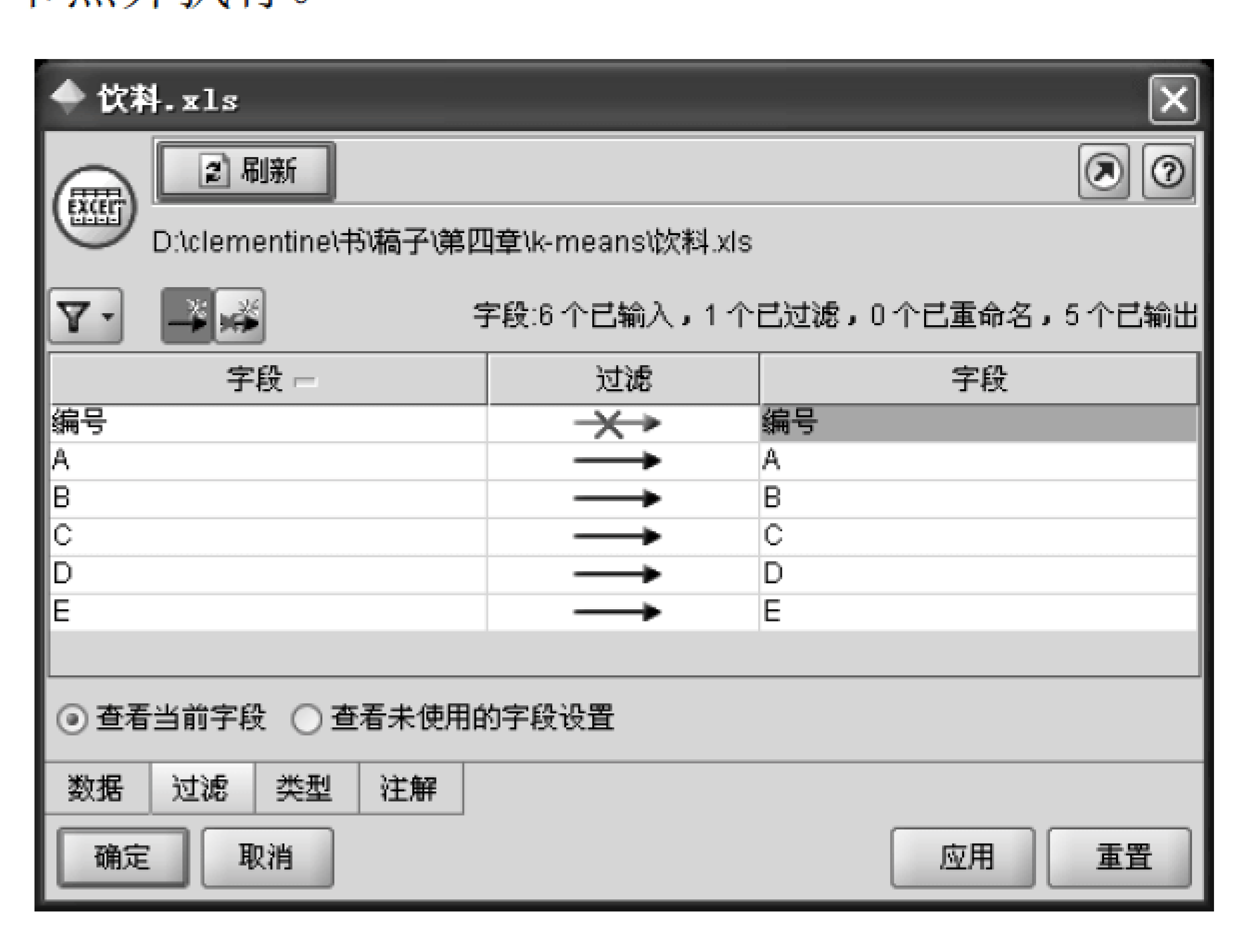


图 4.2　过滤“编号”字段

在数据流区域中添加“类型”节点，并建立从“饮料.xls”节点到“类型”节点的连接。双击“类型”节点，打开“类型”对话框，单击“读取值”按钮，如图 4.3 所示，然后单击“确定”按钮。



图 4.3　设置数据类型

2．设置建模节点

在“建模”标签下选择 K-Means 节点添加到数据流区域，并建立从“类型”节点到 K-Means 节点的连接，双击 K-Means 节点，对该节点进行编辑，如图 4.4 所示。

在“模型”标签下，可以设置下列参数及选项：

模型名称（Model Name）：指定要产生的模型名称，有两个选项：

❑ 自动（Auto）。选择该选项后，模型名称将为 K-Means。这是默认的设置。



图 4.4　设置模型参数

- 自定义（Custom）。选择该选项则可以为节点创建的模型指定用户定义的模型名称。

使用分区数据（Use Partitioned Data）：如果用户定义了分割数据集，选择训练数据集作为建模数据集，并利用测试数据集对模型评价。

聚类数（Specified Number of Clusters）：指定生成聚类的类数。默认值为 5，这里设置为 3。

生成距离字段（Generate Distance Field）：如果选择了这一项，生成模型将包括一个具有每个记录与其所属类群中心距离的字段。

显示聚类相似度（Show Cluster Proximity）：选择该选项以在生成模型的输出结果中包含每个簇的质心与其他质心的距离。

聚类标签（Cluster Display）：指定生成聚类类别字段的格式。类别可以用 String（字符）表示，使用指定的 Label Prefix（标签前缀）（如“cluster1”、“cluster2”），或者用 Number（数字）表示。这里采用默认设置，即聚类标签为“字符串”，标签前缀为“聚类”。

切换到“专家”标签下（如图 4.5 所示），设置 K-Means 节点的高级选项。



图 4.5　K-Means 节点的高级选项

在“专家”标签下，可以设置下列参数及选项：

模式（Mode）：有两个选项，“简单”和“专家”。在“简单”模式下，直接采用默认设置。如果需要改变某些参数设置，则选中“专家”单选按钮。

停止（Stop On）：可以指定训练模型的停止条件。默认（Default）停止条件是迭代 20 次或者差异容忍度<0.000001，其中任一项达到就停止。选择自定义（Custom）可以指定自己的停止条件：

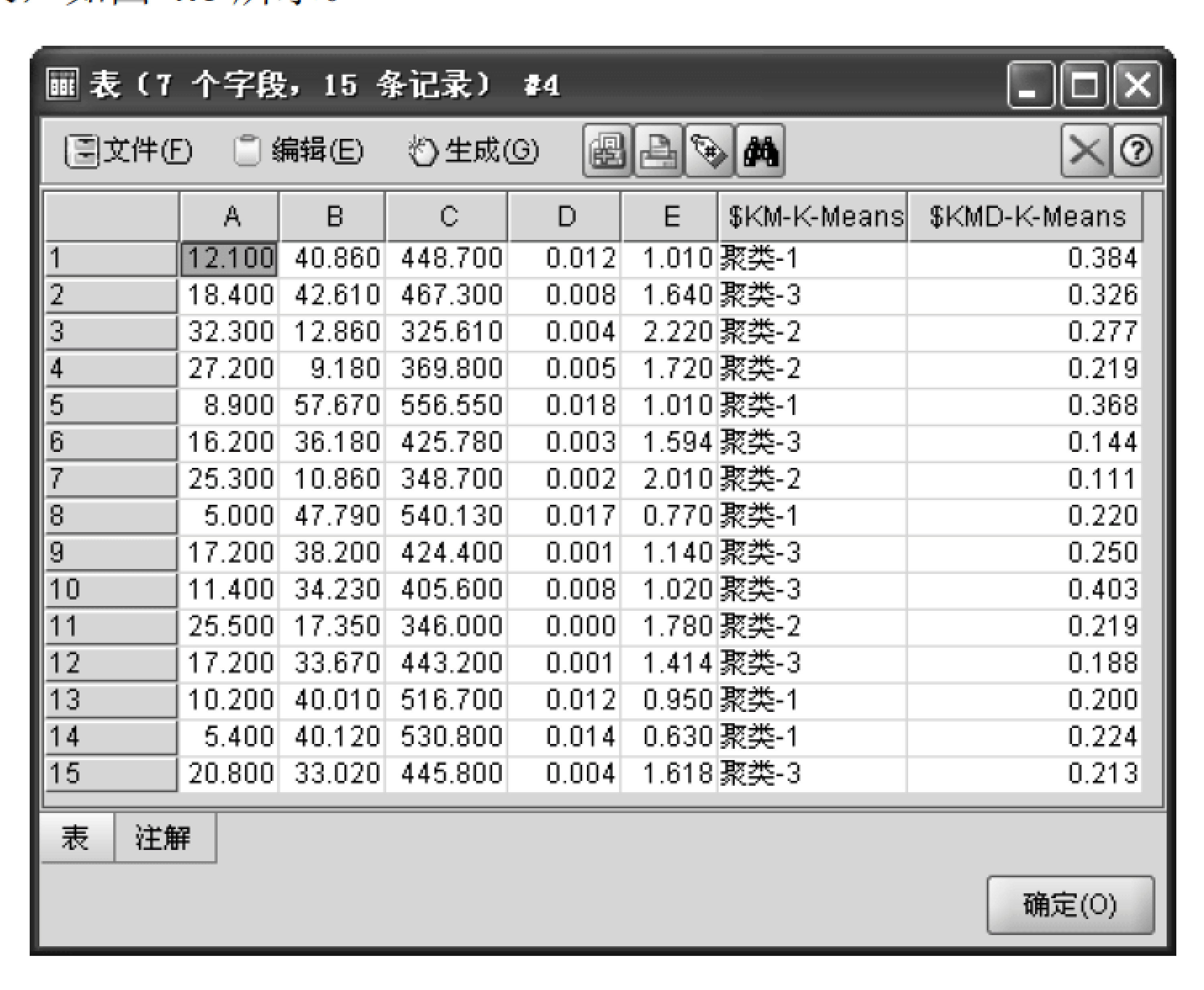
- ❑ 最大迭代数：该选项允许在迭代指定次数后终止训练，这里设置为 10。
- ❑ 差异容忍度：该选项允许在一次迭代中质心之间的最大差异小于指定水平时终止训练。

集合编码值（Encoding Value For Sets）：指定 0～1.0 之间的一个值用于把离散属性重新编码成一组连续型属性。默认值是 0.5 的平方根（大约为 0.707107），值越接近 1.0，离散型属性权重越比连续型属性大。该参数也就是在第 4.2.1 节中阐述的对离散属性进行预处理时需要设置的调节因子 η。

在这些参数设置完毕后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示生成的 K-Means 模型节点。

3. 查看模型的输出结果

将生成的 K-Means 模型节点拖入到数据流区域，并建立从“饮料.xls”节点到 K-Means 模型节点的连接，然后向数据流区域中添加“表”节点，并建立从 K-Means 模型节点到表节点的连接，双击“表”节点，弹出节点的对话框，单击“执行”按钮，即可查看模型的输出结果，如图 4.6 所示。



	A	B	C	D	E	$KM-K-Means	$KMD-K-Means
1	12.100	40.860	448.700	0.012	1.010	聚类-1	0.384
2	18.400	42.610	467.300	0.008	1.640	聚类-3	0.326
3	32.300	12.860	325.610	0.004	2.220	聚类-2	0.277
4	27.200	9.180	369.800	0.005	1.720	聚类-2	0.219
5	8.900	57.670	556.550	0.018	1.010	聚类-1	0.368
6	16.200	36.180	425.780	0.003	1.594	聚类-3	0.144
7	25.300	10.860	348.700	0.002	2.010	聚类-2	0.111
8	5.000	47.790	540.130	0.017	0.770	聚类-1	0.220
9	17.200	38.200	424.400	0.001	1.140	聚类-3	0.250
10	11.400	34.230	405.600	0.008	1.020	聚类-3	0.403
11	25.500	17.350	346.000	0.000	1.780	聚类-2	0.219
12	17.200	33.670	443.200	0.001	1.414	聚类-3	0.188
13	10.200	40.010	516.700	0.012	0.950	聚类-1	0.200
14	5.400	40.120	530.800	0.014	0.630	聚类-1	0.224
15	20.800	33.020	445.800	0.004	1.618	聚类-3	0.213

图 4.6　模型的输出结果

可以看出，在原有数据属性的基础上，输出结果中多了两个属性：$KM-K-Means

和$KMD-K-Means。

$KM-K-Means：表示样本被划分到的那个簇。

$KMD-K-Means：表示样本与其所在簇的质心之间的距离。

例如第一个样本，被划分到了“聚类-1”中，它与“聚类-1”的质心的距离是 0.384。

结果显示，15 个样本被划分为 3 个簇：“聚类-1”包含样本 1、5、8、13、14；“聚类-2”包含样本 3、4、7、11；“聚类-3”包含样本 2、6、9、10、12、15。

4．浏览生成的 K-Means 模型

右击管理器窗口“模型”标签下生成的 K-Means 模型节点，在快捷菜单中选择“浏览”命令，打开 K-Means 对话框，在“模型”标签下会显示划分出来的 3 个聚类，单击“全部展开”按钮，则可以显示每个簇的一些统计信息，如图 4.7 所示。

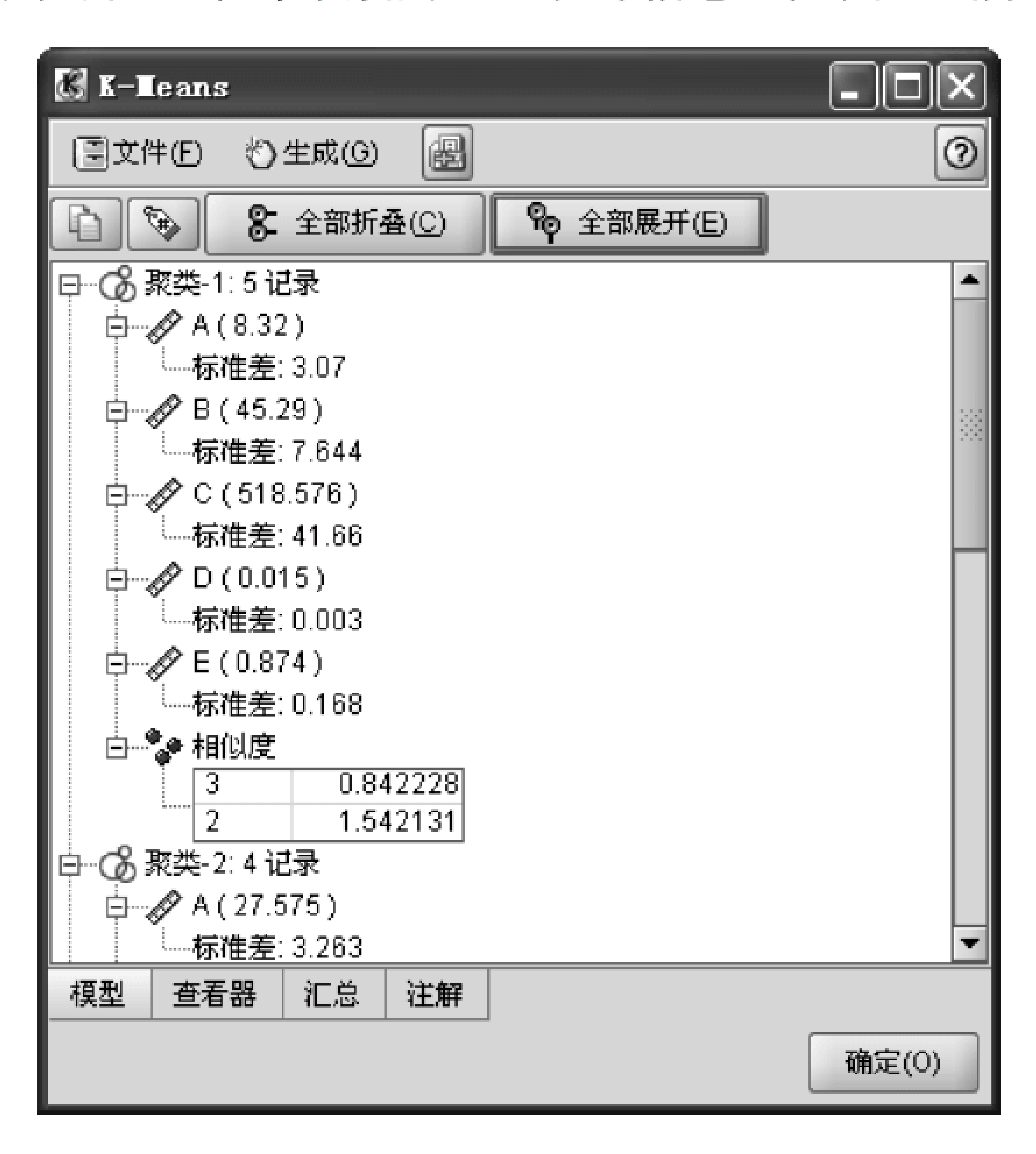


图 4.7　聚类的统计信息

以“聚类-1”为例来说明这些统计信息。“聚类-1”包括 5 个记录（样本），这 5 个记录的属性 A 的均值是 8.32，标准差是 3.07；属性 B 的均值是 45.29，标准差是 7.644；以此类推。在“相似度”一栏中，显示了“聚类-1”与“聚类-3”的距离是 0.842228，与“聚类-2”的距离是 1.542131，之所以会计算相似度，是因为在图 4.4 中设置模型参数时选中了“显示聚类相似度”复选框。

单击“查看器”标签，可以以图表的形式来显示模型的统计信息以及各个属性在各簇中的分布信息，如图 4.8 所示。

默认情况下，各个簇显示在横轴，各属性显示在纵轴。

图中的前几列用来显示各个簇的统计信息，以图例的形式来表示各属性在每个簇中的平均值。各个簇按照包含样本数量的多少降序排列（例如在图 4.8 中，聚类-3 包含了

6 个样本，样本数量最多，所以排列在最前面）。

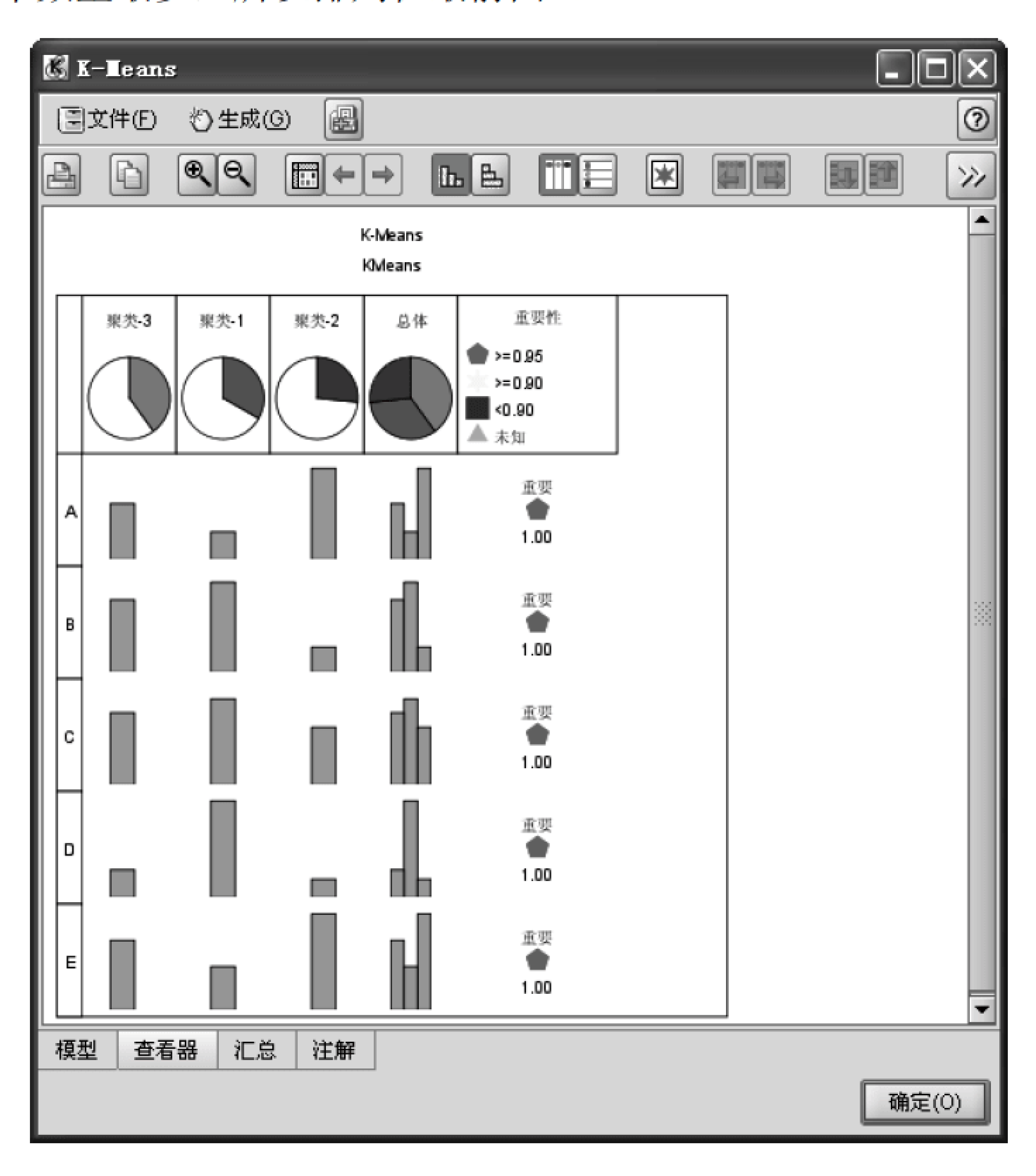


图 4.8　模型的“查看器”标签

“总体”这一列其实就是将前面几列各个簇的图例集中在一起显示，起到一种比较和对照的作用。注意，“总体”这一列并不是默认显示的。如果需要显示“总体”列，需要在扩展对话框中（单击对话框右上角的»按钮即可打开扩展对话框）选择“显示总结果”。

“重要性”这一列则显示了每个属性对于建立模型的重要程度。它用一个 0～1 之间的数值来表示，其数值等于 1–p（p 是用 t 检验或者卡方检验得到的显著度水平。如果是连续属性，用 t 检验；如果是离散属性，则用卡方检验）。

另外，打开扩展窗口，通过对一些选项的选择，还可以实现单独显示某个聚类、用文字形式来显示统计信息等功能，如图 4.9 所示。

通过图 4.9 所示的统计图，通常可以分析两类问题：其一，分析某个属性在所有簇中的分布情况；其二，分析某个簇中各个属性的分布情况。

最后，在窗口的“汇总”标签中，会显示关于本次训练过程中的一些概要信息，包括“分析”、“字段”、“构建设置”、“训练概要”等 4 个栏目，如图 4.10 所示。

在“分析”一栏中，展示由模型建立的聚类方法的相关信息。给出聚类数目及其迭代过程。例如从图 4.10“分析”一栏中可以看到，聚类树为 3，算法进行了 3 次迭代。在这 3 次迭代中，$\max_{J}\|C_j(t)-C_j(t-1)\|$的值逐渐减小，第 3 次迭代时为 0，满足了算法停止准则，算法终止。



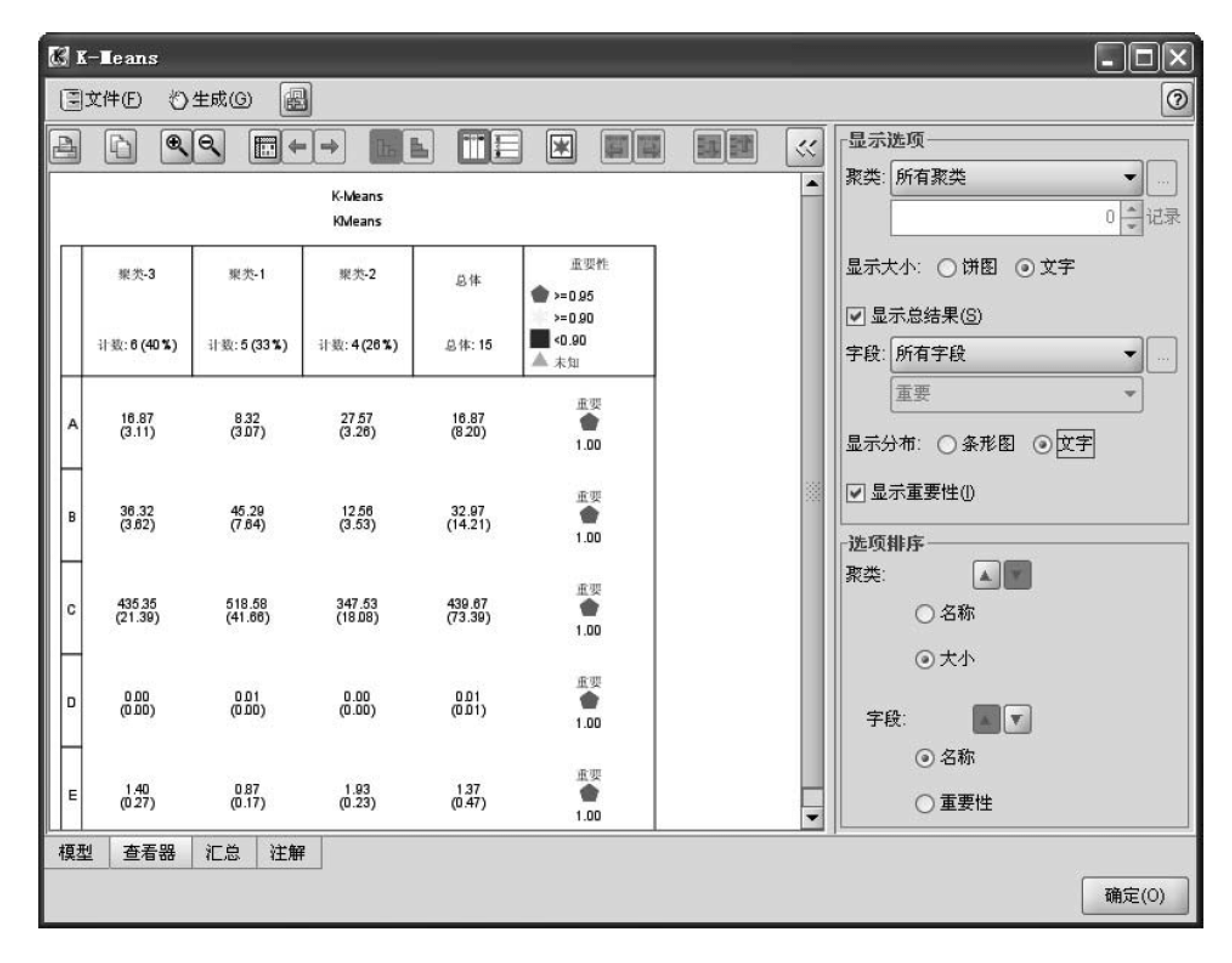


图 4.9　以文字形式显示统计信息

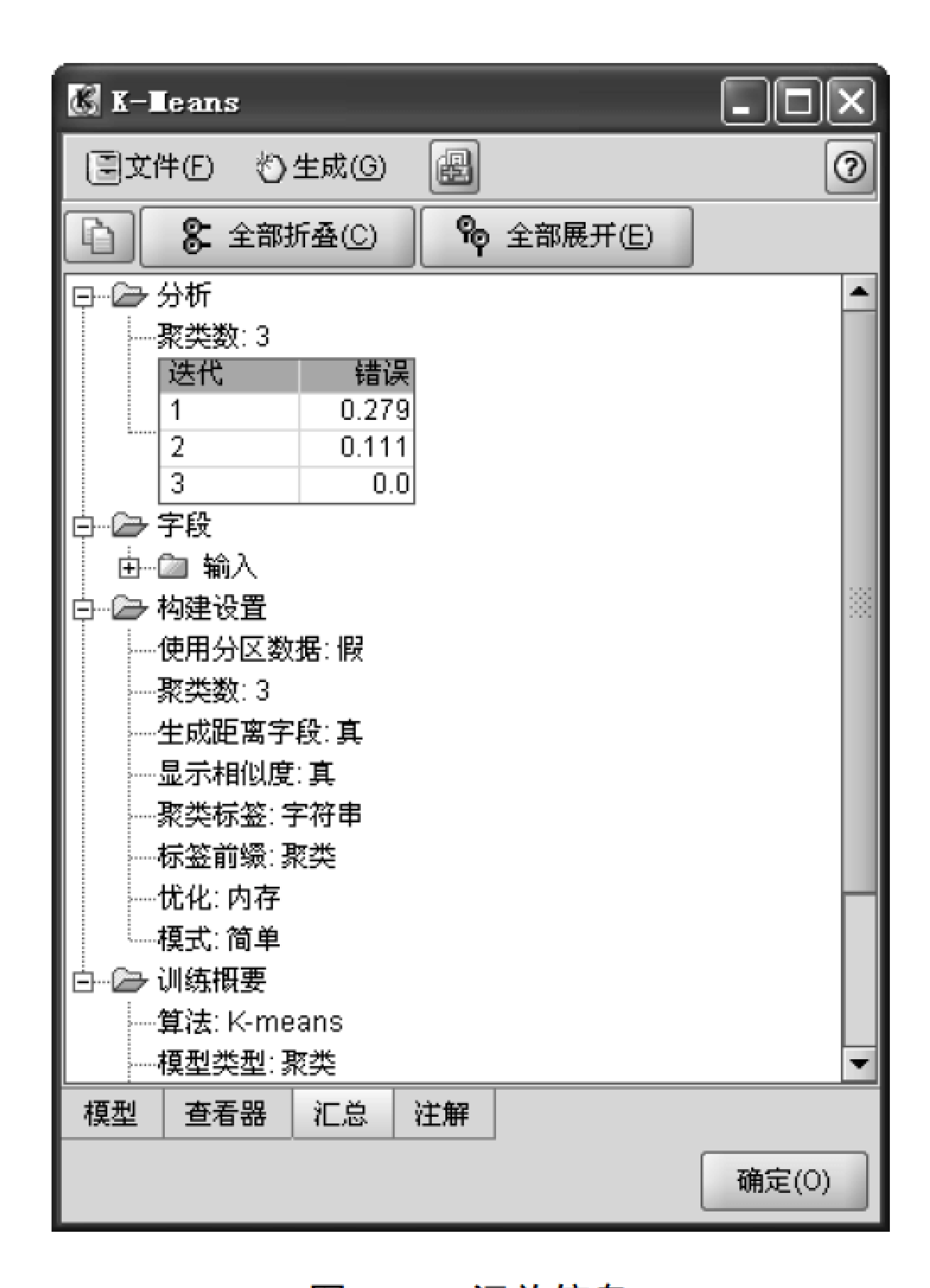


图 4.10　汇总信息

4.3 TwoStep 算法

TwoStep（两步聚类）是一种分两步进行的聚类方法，用于处理非常大的数据集，可以处理连续属性和离散属性。它只需遍历数据集一次。整个算法包括以下两个步骤：

第一步：预聚类。遍历一次数据，在这个过程中把原始数据压缩成易处理的子类集。这一步骤是通过 BIRCH 算法（Zhang, Ramakrishnon, Livny,1996）构造 CF 树来实现的。算法一个一个地扫描样本，并决定是否将当前样本指派到已经生成的某个聚类中，或是生成一个新的聚类（根据距离法则）来存放当前样本。

第二步：聚类。使用层次聚类方法，将小的聚类逐渐合并成越来越大的聚类，这一过程不需要再次遍历数据。层次聚类的好处是不要求提前选择聚类数。许多层次聚类从单个记录开始聚类，逐步合并成更大的类群。

在 TwoStep 中，算法可以自动选择聚类的个数。

4.3.1 构建 CF 树

1. CF

一个聚类是由一些相近的数据样本构成的，为了简化问题的描述和数据处理过程，可以用聚类的特征（Cluster Feature，CF）来代表这个聚类（而无须涉及该聚类中的所有样本本身）。

给定一个聚类 $\{\overrightarrow{X_i}\}$，包括 N 个 d 维（即 d 个属性）的数据样本，其中 i=1,2,⋯,N。用一个三元组来表示它的聚类特征，并以 $\mathrm{CF}=(N,\overrightarrow{\mathrm{LS}},\mathrm{SS})$ 这样的表达式来表示，称为 CF 条目（CF entry），这里 N 是聚类中的样本个数，$\overrightarrow{\mathrm{LS}}$ 是 N 个样本的线性和：$\overrightarrow{\mathrm{LS}}=\sum_{i=1}^{N}\overrightarrow{X_i}$，SS 是 N 个样本的平方和：$\mathrm{SS}=\sum_{i=1}^{N}\overrightarrow{X_i}^2$。

设 $\mathrm{CF}_1=(N_1,\overrightarrow{\mathrm{LS}_1},\mathrm{SS}_1)$ 和 $\mathrm{CF}_2=(N_2,\overrightarrow{\mathrm{LS}_2},\mathrm{SS}_2)$ 是两个子聚类的 CF 条目，那么把这两个子聚类合并后的聚类的 CF 条目为：

$$\mathrm{CF}_1+\mathrm{CF}_2=(N_1+N_2,\overrightarrow{\mathrm{LS}_1}+\overrightarrow{\mathrm{LS}_2},\mathrm{SS}_1+\mathrm{SS}_2)$$

CF 条目是简洁的（不必存储所有的数据点），但同时又是精确的（利用 CF 条目足以计算各种距离以进行聚类分析）。

2. CF 树

CF 树是一棵形如图 4.11 所示的树。它有两个参数，分支因子（非叶子节点的分支因子是 B，叶子节点的分支因子是 L）和阈值 T。每个节点由一些 CF 条目构成。

每个非叶子节点最多包含了 B 个条目，每个条目的形式是[CF_i,child_i]，i=1,2,⋯,B，child_i 是一个指针，指向第 i 个孩子节点，CF_i 是这个孩子节点代表的那个子聚类的 CF 条

目。所以一个非叶子节点代表了一个子聚类，这个子聚类又分为多个子聚类（孩子节点），这些关系都是用 CF 条目来表示的。

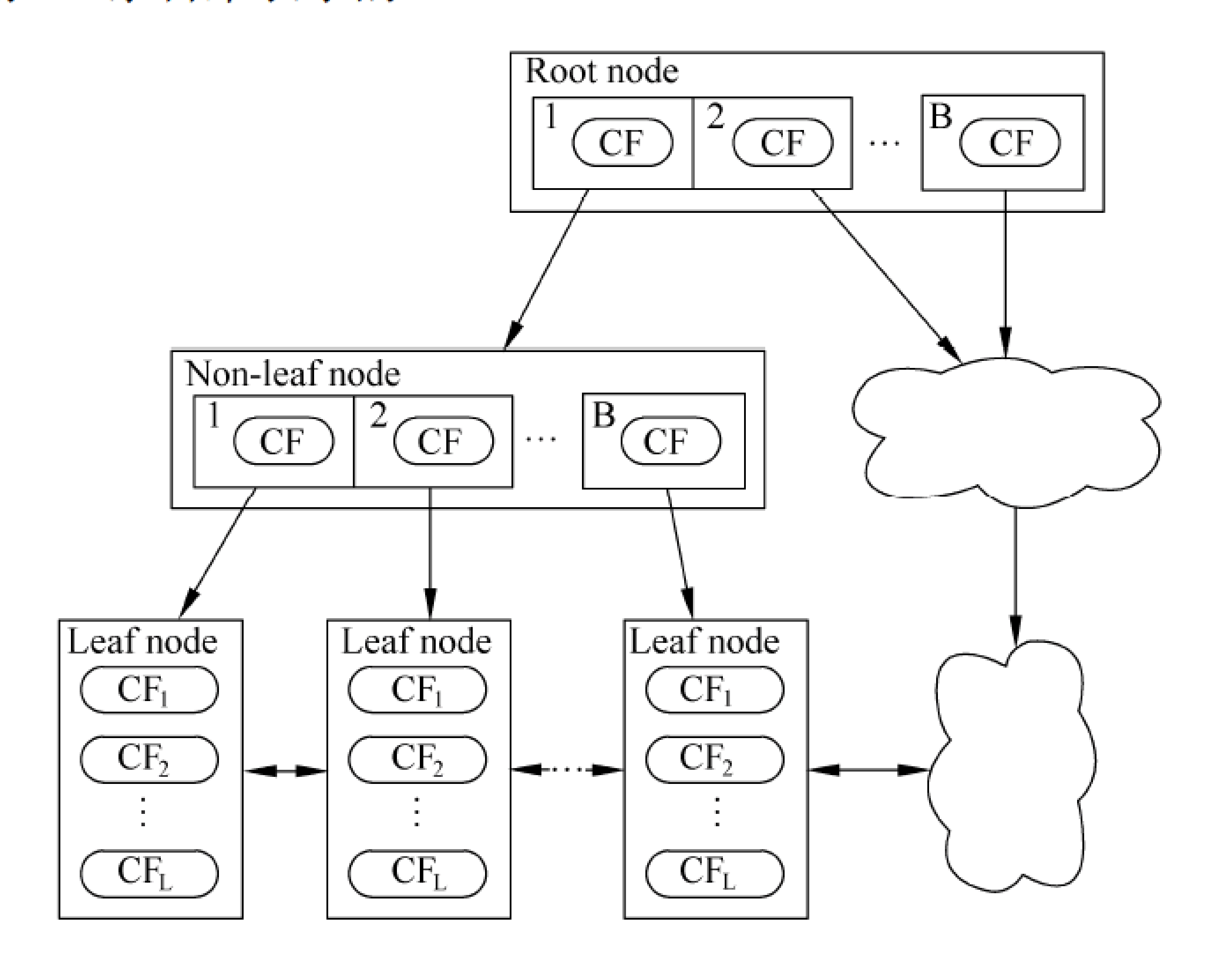


图 4.11　CF 树实例

一个叶子节点最多包含 L 个条目，每个条目代表一个子聚类。一个叶子节点同时也代表着一个聚类，这个聚类又是由多个子聚类组成，每个子聚类用各自的条目来表示。但是，一个叶子节点的所有条目都必须满足一个阈值要求，这就和阈值 T 有关：叶子节点条目的直径（或者有时采用半径）必须小于 T。给定由 N 个 d 维的数据样本组成的数据集 $\{\overrightarrow{X}_i\}$，其中 i=1,2,⋯,N，质心为 $\overrightarrow{X}_0$，半径为 R，直径为 D。这些参数的定义如下：

$$\overrightarrow{X}_0=\frac{\sum_{i=1}^{N}\overrightarrow{X}_i}{N}$$

$$R=\sqrt{\frac{\sum_{i=1}^{N}(\overrightarrow{X}_i-\overrightarrow{X}_0)^2}{N}}$$，是簇内各样本点到质心的平均距离。

$$D=\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(\overrightarrow{X}_i-\overrightarrow{X}_j)^2}{N(N-1)}}$$，是簇内任意两样本点间的距离的平均值。R 和 D 是两种簇内“紧密度”的度量方法。

显然，树的规模大小是与阈值 T 相关的。T 越大，一个叶子节点条目就可以包含更多的样本点，那么树的总的规模就越小。

3．生成 CF 树

CF 树是在遍历数据集的过程中不断添加、更新条目及分裂节点来形成的。根据第一个样本即可建立根节点及相应的条目，之后逐个地将后续的样本根据距离最小的原则指

派到 CF 树中。

对于一个样本 Ent，首先从根节点开始，沿着 CF 树而下，选择和这个样本距离最小的叶子节点；然后在这个叶子节点里寻找最近的条目（一个条目代表一个子聚类），例如 L_i，测试一下 L_i 在吸收 Ent 之后是否违背阈值条件 T。如果没有违背阈值条件，就更新 L_i 的条目，以反映出这一变化（增加了一个样本）。如果违背了阈值条件，就给这个叶子节点增加一个新的条目，来代表 Ent。但如果该叶子节点已经没有空间来存放一个新的条目（一个叶子节点最多只能有 L 个条目），必须分裂这个叶子节点。分裂的方法是：选择两个距离最远的条目作为种子（两个叶子节点），然后根据距离最小原则，把剩下的条目重新分配到这两个叶子节点中。

在把 Ent 添加到叶子节点后，要更新从根节点到这个叶子节点的路径上的所有非叶子节点的条目信息。如果叶子节点没有分裂，则相对简单，更新相关的 CF 条目即可。如果叶子节点分裂了，则还要给这个叶子节点的父节点增加一个条目，来代表这个新的叶子节点。如果父节点还有空间来存放这个条目（父节点最多能存放 B 个条目），那么给这个父节点增加条目即可。如果没有空间，还需要分裂这个父节点，甚至一直分裂到根节点。

另外，树的规模是受到限制的，由分支因子（B 和 L）和阈值 T 来决定。如果树上已经没有了空间来存放新的样本时，可以通过增大 T 的值并在已有的 CF 树基础上重建 CF 树。重建后的 CF 树规模就小一些，这样就有了多余的空间来存放新的样本。

以上过程不断重复，直到完成一次数据遍历。

4.3.2 聚类

在得到一棵 CF 树之后，每个叶子节点的每个条目都代表了一个聚类，下面将这些聚类根据距离最小的原则进行合并，得到数量适当的聚类。这里 TwoStep 利用了凝聚的层次聚类方法。

1. 凝聚的层次聚类

层次的聚类方法根据数据集组成一棵聚类的树。根据层次分解是自底向上还是自顶向下形成，层次的聚类方法可以进一步分为凝聚的（agglomerative）和分裂的（divisive）层次聚类。一个纯粹的层次聚类方法的聚类质量受限于如下特点：一旦一个合并或分裂被执行，就不能修正。

凝聚的层次聚类：这种自底向上的策略首先将每个样本作为一个簇，然后合并这些原子簇为越来越大的簇，直到所有的样本都在一个簇中，或者某个终结条件被满足。绝大多数层次聚类方法属于这一类，它们只是在簇间相似度的定义上有所不同。它的基本方法是：给定要聚类的 n 个簇，计算它们的 $n \times n$ 的距离矩阵，找到最接近的两个类并合并成一类，于是总的类数少了一个，然后重新计算新的类与所有旧类之间的距离，再选择最接近的两个类合并，以此类推，直到最后合并成一个类（或者满足某个停止条件）为止。

分裂的层次聚类：这种自顶向下的策略与凝聚的层次聚类相反，它首先将所有样本

置于一个簇中，然后逐渐细分为越来越小的簇，直到每个样本自成一簇，或者达到了某个终结条件，例如达到了某个希望的簇数目，或者两个最近的簇之间的距离超过了某个阈值。

TwoStep 以每个叶子节点的每个条目为原子簇，利用凝聚的层次聚类方法，对这些簇不断进行合并。

2. 距离的度量

有许多种方法来度量两个聚类之间的距离。TwoStep 聚类方法用对数似然距离度量方法，来处理连续属性和离散属性。它是一个基于概率的距离。两个聚类之间的距离大小，要看它们合并之后对数似然值下降了多少。在计算对数似然值时，假设连续属性的取值为正态分布，离散属性的取值为多项分布。同时，假设各属性之间不相关，样本之间也不相关。聚类 i 和聚类 j 之间的距离为：

$$d(i,j)=\xi_i+\xi_j-\xi_{<i,j>}$$

其中，

$$\xi_v=-N_v\left(\sum_{k=1}^{K^A}\frac{1}{2}\log(\hat{\sigma}_k^2+\hat{\sigma}_{vk}^2)+\sum_{k=1}^{K^B}\hat{E}_{vk}\right),\quad \hat{E}_{vk}=-\sum_{l=1}^{L_k}\frac{N_{vkl}}{N_v}\log\frac{N_{vkl}}{N_v}$$

在这些表达式中：

K^A 是输入属性中连续属性的个数；

K^B 是输入属性中离散属性的个数；

L_k 是第 k 个离散属性的不同取值的个数；

N_v 是聚类 v 中的样本个数；

N_{vkl} 是聚类 v 中第 k 个离散属性取值为 l 的样本的数量；

$\hat{\sigma}_k^2$ 是第 k 个连续属性的全部样本值的估计方差；

$\hat{\sigma}_{vk}^2$ 是在聚类 v 中第 k 个连续属性的全部样本值的估计方差；

$<i,j>$ 代表由聚类 i 和聚类 j 组合而成的聚类。

如果在计算 ξ_v 时忽略 $\hat{\sigma}_k^2$，那么聚类 i 和聚类 j 之间的距离就是两个聚类合并前后的对数似然值的差。$\hat{\sigma}_k^2$ 在式中的作用是避免遇到 $\hat{\sigma}_{vk}^2=0$ 的情况（当聚类 v 中只有一个样本时，$\hat{\sigma}_{vk}^2=0$，使得自然对数无意义）。

3. 确定聚类数

在 K-Means 算法中，最终的聚类树是由用户事先指定的。但在 TwoStep 算法里，由算法来自动确定究竟将数据集中的样本聚为多少类。

由于采用凝聚的层次聚类方法，给定要聚类的 n 个簇，每合并两个簇，聚类的数量就少 1，因此在聚类的过程中就形成了一个聚类数量序列：$n,(n-1),\cdots,3,2,1$，代表了 n 种聚类方案。

为了自动地确定聚类的数量，TwoStep 算法采用了一个“两阶段”方法。

（1）第一阶段：计算聚类数的初步估计值

对每一种聚类方案，计算它的贝叶斯信息判别式（Bayes Information Criterion，BIC）。

例如，有 J 个聚类的划分方案的 BIC 为：

$\mathrm{BIC}(J)=-2\sum_{j=1}^{J}\xi_j+m_J\log(N)$，其中，$m_J=J\{2K^A+\sum_{k=1}^{K^B}(L_K-1)\}$，其他参数定义如前。

令 $d\mathrm{BIC}(J)=\mathrm{BIC}(J)-\mathrm{BIC}(J+1)$，表示 J 个聚类的方案和（J+1）个聚类的方案之间的差别。那么，计算 $R_1(J)=\dfrac{d\mathrm{BIC}(J)}{d\mathrm{BIC}(1)}$，表示 $d\mathrm{BIC}(J)$ 相对于 $d\mathrm{BIC}(1)$ 的改变程度。如果 $d\mathrm{BIC}(1)<0$，那么聚类树就设置为 1（第二阶段省略）。否则，聚类数的初步估计值 k 就是使得 $R_1(J)<0.04$ 成立的最小的 J。

（2）第二阶段：确定聚类数

根据第一阶段确定的 k 值，计算 $R_2(k)=\dfrac{d_{\min}(C_k)}{d_{\min}(C_{k+1})}$。其中，$C_k$ 表示聚类数为 k 的划分方案，$d_{\min}(C_k)$ 表示该方案中距离最小的两个簇之间的距离。C_{k+1} 表示聚类数为 k 的划分方案，$d_{\min}(C_{k+1})$ 表示该方案中距离最小的两个簇之间的距离。

同理，可计算出 $R_2(k–1), R_2(k–2),\cdots, R_2(2)$，得到一个 R_2 比值序列。从中选出两个最大的比值 $R_2(i)$ 和 $R_2(j)$，$R_2(i)>R_2(j)$。如果 $R_2(i)$ 比 $R_2(j)$ 的 1.15 倍还要大，那么最终的聚类数就等于 i；否则选择 i 和 j 中的大者（即选择聚类数多者）作为最终的聚类数。

4.3.3 在 Clementine 中应用 TwoStep

本节在 Clementine 中用 TwoStep 算法对第 4.2.3 节中的液体饮料聚类分析实例进行数据挖掘，可以对比两种聚类方法产生的不同结果。样本数据集在表 4-3 中。

用 TwoStep 算法对液体饮料进行聚类分析的数据流如图 4.12 所示。

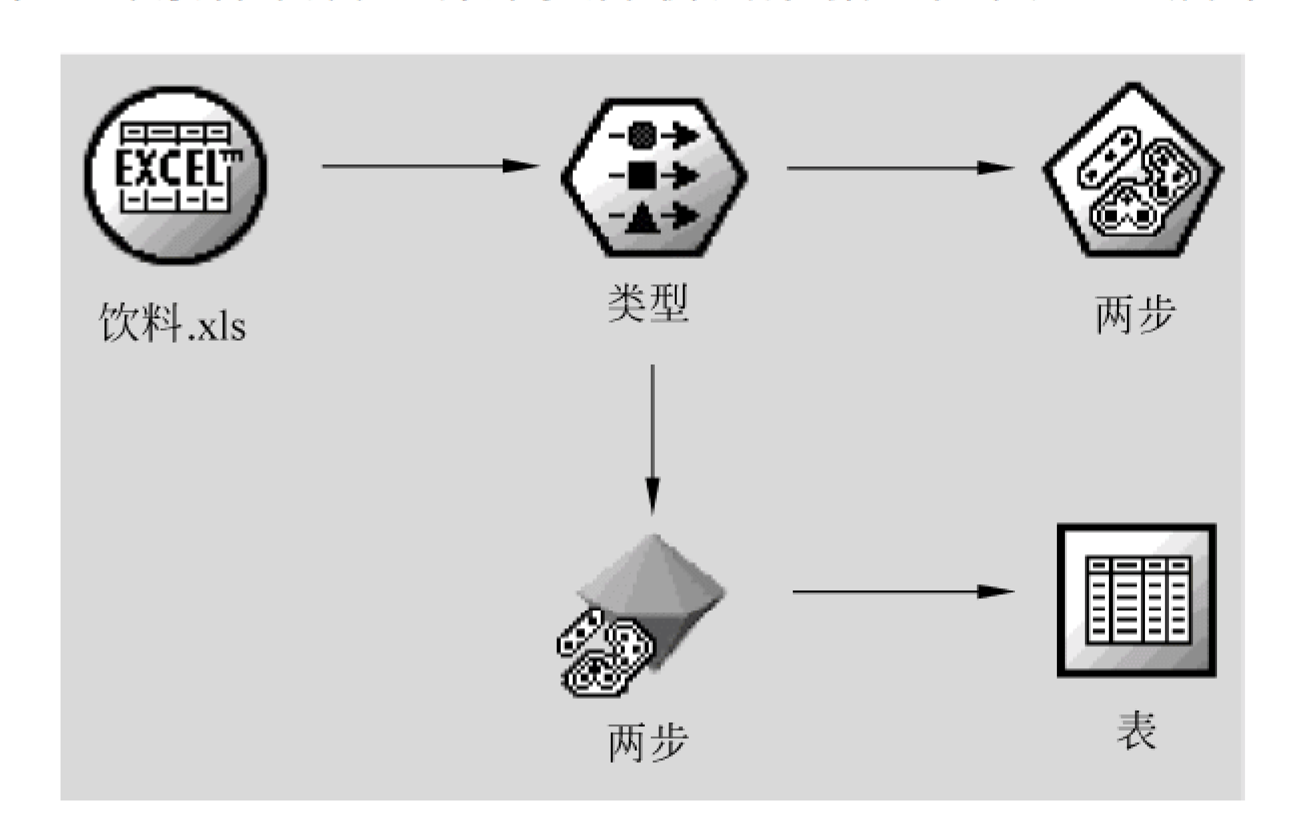


图 4.12　TwoStep 聚类分析数据流

首先在数据流区域添加 Excel 数据源节点并导入“饮料.xls”文件，在该节点将“编号”属性过滤。然后添加“类型”节点，并读取值。

在数据流区域添加“两步”节点，并建立从“类型”节点到“两步”节点的连接，然后编辑“两步”节点，如图 4.13 所示。



图 4.13 编辑“两步”节点

图 4.13 中的各选项说明如下：

模型名称（Model Name）：指定要产生的模型名称。

- 自动（Auto）：选择该选项后，模型名称将为 TwoStep。这是默认的设置。
- 自定义（Custom）：选择该选项可以为节点创建的模型指定用户定义的模型名称。

使用分区数据（Use Partitioned Data）：如果用户定义了分割数据集，选择训练数据集作为建模数据集，并利用测试数据集对模型评价。

标准化数值字段（Standardize Numeric Fields）：默认设置下，两步聚类会统一把所有数值型输入字段标准化为均值为 0，方差为 1。如果要保留数据的原始度量，取消选择该项。

排除离群值（Exclude Outliers）：如果选择了该项，输入字段具有异常值的记录将从分析中自动排除。这样能够避免此类个案歪曲结果。

聚类标签（Cluster Label）：指定生成模型聚类类别字段格式。类别可以用字符串（String）表示，使用指定的标签前缀（Label prefix）（如“cluster 1”、“cluster 2”），或者用数值（Number）表示。

自动计算聚类数（Automatically Calculate Number of Clusters）：两步聚类能够迅速分析大量聚类方案，并为训练集选出最优类别数。通过设置最大值（Maximum）和最小值（Minimum）聚类数指定聚类数范围。两步聚类使用两阶段过程来决定最优聚类数。在第一阶段，根据贝叶斯信息判别式（BIC）的变化添加的类别数来选择模型的类别上界。在第二阶段，找出所有类别数小于最小-BIC 方案的模型的聚类间最小距离变化。距离变化最大处用于区分最终的聚类模型。

指定聚类数（Specify Number of Clusters）：如果知道模型中所包含的类别数，选择该选项并输入类别数。

按照图中设置完毕后，单击“执行”按钮，即可在管理器窗口得到生成的两步模型。

将生成的两步模型拖入到数据流区域，并建立从“类型”到该节点的连接，然后在模型节点后添加“表”节点，执行“表”节点，即可看到聚类的结果，如图 4.14 所示。



	A	B	C	D	E	$T-两步
1	12.100	40.860	448.700	0.012	1.010	聚类-2
2	18.400	42.610	467.300	0.008	1.640	聚类-1
3	32.300	12.860	325.610	0.004	2.220	聚类-1
4	27.200	9.180	369.800	0.005	1.720	聚类-1
5	8.900	57.670	556.550	0.018	1.010	聚类-2
6	16.200	36.180	425.780	0.003	1.594	聚类-1
7	25.300	10.860	348.700	0.002	2.010	聚类-1
8	5.000	47.790	540.130	0.017	0.770	聚类-2
9	17.200	38.200	424.400	0.001	1.140	聚类-1
10	11.400	34.230	405.600	0.008	1.020	聚类-2
11	25.500	17.350	346.000	0.000	1.780	聚类-1
12	17.200	33.670	443.200	0.001	1.414	聚类-1
13	10.200	40.010	516.700	0.012	0.950	聚类-2
14	5.400	40.120	530.800	0.014	0.630	聚类-2
15	20.800	33.020	445.800	0.004	1.618	聚类-1

图 4.14　TwoStep 聚类结果

由结果中的“$T-两步”列可知，TwoStep 算法将 15 种饮料自动划分为两类：聚类-1（含 9 个样本）和聚类-2（含 6 个样本）。

右击管理器窗口“模型”标签下生成的两步模型节点，在快捷菜单中选择“浏览”命令，打开“两步”对话框，可分别在“模型”、“查看器”和“汇总”标签下浏览生成模型的各项统计信息，浏览模型的具体方法见第 4.2.3 节中的相关阐述。

第 5 章　关 联 规 则

5.1 关联规则概述

5.1.1　关联规则的定义

数据关联是数据库中存在的一类重要的可被发现的知识。若两个或多个变量的取值之间存在某种规律性，就称为关联。这种关联体现了事物及事物之间的规律，掌握这些规律，可以对人们的行为进行有效地指导。因此，关联规则挖掘成为了数据挖掘中的一项重要内容。

关联规则挖掘的一个典型例子是购物篮分析。市场分析员要从大量的数据中发现顾客放入其购物篮中的不同商品之间的关系。如果顾客买牛奶，他也购买面包的可能性有多大？什么商品组或集合顾客多半会在一次购物时同时购买？例如，买牛奶的顾客有80%也同时买面包，或买铁锤的顾客中有 70%的人同时也买铁钉，这就是从购物篮数据中提取的关联规则。分析结果可以帮助经理设计不同的商店布局。一种策略是：经常一块购买的商品可以放近一些，以便进一步刺激这些商品一起销售，例如，如果顾客购买计算机又倾向于同时购买财务软件，那么将硬件摆放离软件陈列近一点，可能有助于增加两者的销售。另一种策略是：将硬件和软件放在商店的两端，可能诱发购买这些商品的顾客一路挑选其他商品。

“啤酒与尿布”的故事，就是一个经典的关联规则挖掘案例。沃尔玛超市的数据仓库里集中了其各门店的详细原始交易数据。在这些原始交易数据的基础上，技术人员利用数据挖掘方法对这些数据进行分析和挖掘。一个意外的发现是：跟尿布一起购买最多的商品竟是啤酒！经过大量实际调查和分析，揭示了一个隐藏在“啤酒与尿布”背后的美国人的一种行为模式：在美国，一些年轻的父亲下班后经常要到超市去买婴儿尿布，而他们中有 30%～40%的人同时也为自己买一些啤酒。于是，啤酒和尿布被摆在一起出售。

关联可分为简单关联、时序关联、因果关联。关联分析的目的是找出数据库中隐藏的关联，并以规则的形式表达出来，这就是关联规则。

有时并不知道数据库中数据的关联函数，即使知道也是不确定的，因此关联分析生成的规则带有置信度（可信度）。关联规则挖掘发现大量数据中项集之间有趣的关联或相关联系。

Agrawal 在 1993 年首先提出了挖掘顾客交易数据库中项集间的关联规则问题，以后诸多的研究人员对关联规则的挖掘问题进行了大量的研究。他们的工作包括对原有的算法进行优化，如引入随机采样、并行的思想等，以提高算法挖掘规则的效率；对关联规则的应用进行推广。关联规则挖掘成为数据挖掘中一个重要的研究方向。

5.1.2 关联规则的基本概念

假如一个超市的销售记录（如表 5-1 所示）由许多样本组成，每个样本由若干个属性来确定。

表 5-1 销售记录数据集

编号	牛奶	果冻	啤酒	面包	花生酱
T_1	1	1	0	0	1
T_2	0	1	0	1	0
T_3	0	1	1	0	0
T_4	1	1	0	1	0
T_5	1	0	1	0	0
T_6	0	1	1	0	0
T_7	1	0	1	0	0
T_8	1	1	1	0	1
T_9	1	1	1	0	0

表中共有 9 个样本，每个样本记录了一次购物的商品（属性值取“1”表示已购买，“0”表示未购买），这里把一个样本称为一个“**事务**”（Transaction）。每个事务由多个属性来确定，这里的属性称为“**项**”（Item），多个项组成的集合称为“**项集**”（Itemset）。根据项集中包含的项的数量，项集可以是 1-项集，2-项集，或者 k-项集。例如{牛奶}、{啤酒}都是 1-项集；{牛奶，果冻}是 2-项集；{啤酒，面包，牛奶}是 3-项集。可以看出，每个事务其实就是一个项集，例如事务 T_1 就是项集{牛奶，果冻，花生酱}。

给定一个含有 n 个事务的数据库 $D=\{t_1, t_2,\cdots,t_n\}$，有 m 个属性，这 m 个属性组成的项集为 $I=\{i_1,i_2,\cdots,i_m\}$。那么其中的每个事务 t 都是一个项集，且 $t \subseteq I$ 。设 A 是一个项集，当 $A \subseteq t$ 时称“事务 t 包含 A”。

关联规则是形如 $X \Rightarrow Y$ 的蕴含式，其中 X 和 Y 是项集，且 $X \subset I$ ，$Y \subset I$ ，$X \cap Y = \varnothing$ 。X 称为规则**前项**（或者前件，antecedent），Y 称为规则**后项**（或者后件，consequent）。

关联规则 $X \Rightarrow Y$ 的**支持度** s 是数据库中包含 $X \cup Y$ 的事务占全部事务的百分比，它是概率 $P(X \cup Y)$ ，记做 $\text{support}(X \Rightarrow Y) = P(X \cup Y)$ 。

关联规则 $X \Rightarrow Y$ 的**置信度** c 是包含 $X \cup Y$ 的事务数与包含 X 的事务数的比值，它是条件概率 $P(Y \mid X)$ ，记做 $\text{confidence}(X \Rightarrow Y) = P(Y \mid X)$ 。

在进行关联规则挖掘之前，由用户预先定义**最小支持度阈值**（min_sup）和**最小置信度阈值**（min_conf）。我们一般并不是关心所有的规则，而只对那些重要的规则感兴趣，它们都是支持度大于等于 min_sup、置信度大于等于 min_conf 的规则，这些规则称为“**强规则**”。

如果某个项集的支持度大于等于设定的最小支持度阈值 min_sup，称这个项集为“**频繁项集**”（也称为“大项集”，LargeItemsets），所有的“频繁 k-项集”组成的集合通常记做 L_k。

因此，关联规则挖掘过程主要包含两个阶段：第一阶段先从数据集中找出所有的频

繁项集，它们的支持度均大于等于最小支持度阈值 min_sup，第二阶段由这些频繁项集产生关联规则，计算它们的置信度，然后保留那些置信度大于等于最小置信度阈值 min_conf 的关联规则。

根据关联规则中项的类别不同，关联规则可以分为布尔型和数值型。布尔型关联规则处理的值都是离散的、种类化的属性，它显示了这些属性之间的关系；而数值型关联规则可以和多维关联或多层关联规则结合起来，对数值型属性进行处理，将其进行动态的分割，或者直接对原始的数据进行处理，当然数值型关联规则中也可以包含种类变量。例如：性别=“女”⇒职业=“秘书”，是布尔型关联规则；性别=“女”⇒avg（收入）=2300，涉及的收入是数值类型，所以是一个数值型关联规则。

关联规则也可以分为单层关联规则和多层关联规则。在单层的关联规则中，所有的变量都没有考虑到现实的数据是具有多个不同的层次的；而在多层的关联规则中，对数据的多层性已经进行了充分的考虑。例如：IBM 台式机⇒Sony 打印机，是一个细节数据上的单层关联规则；台式机⇒Sony 打印机，是一个较高层次和细节层次之间的多层关联规则。

另外，根据规则中涉及的数据的维数，关联规则可以分为单维的和多维的。在单维的关联规则中，只涉及数据的一个维，如用户购买的物品；而在多维的关联规则中，要处理的数据将会涉及多个维。换句话说，单维关联规则是处理单个属性中的一些关系；多维关联规则是处理各个属性之间的某些关系。例如：啤酒=>尿布，这条规则只涉及用户购买的物品；性别=“女”⇒职业=“秘书”，这条规则就涉及两个字段的信息，是两个维上的一条关联规则。

关联规则挖掘通常比较适用于属性值为布尔型的事务数据库。如果原始数据库中的属性值是连续型，则在关联规则挖掘之前应该进行适当的数据离散化，数据的离散化是数据挖掘前的重要环节，离散化的过程是否合理将直接影响关联规则的挖掘结果。

5.1.3 关联规则挖掘算法

关联规则挖掘算法是关联规则挖掘研究的主要内容，迄今为止已提出了许多高效的关联规则挖掘算法，并不断有学者提出对这些算法的改进方案，以提高算法的性能。关联规则算法的效率通常针对需要进行的数据库遍历次数和必须进行计数的项集的最大数目来讨论。

1. 广度优先算法

广度优先算法自底向上地搜索整个搜索空间，首先生成候选项集，然后测试产生的项集是否是频繁项集。

经典的广度优先算法是 Agrawal 等提出的 Apriori、AprioriTid 和 AprioriHybrid 算法。

Apriori 算法利用了频繁项集的非单调性：如果一个 k-项集是非频繁项集，那么它的超集一定是非频繁项集。因此，Apriori 算法只用频繁 k-项集构造(k+1)-项集来作为候选项集。但是 Apriori 需要多次遍历数据库，I/O 开销大。

AprioriTid 算法通过把事务替换成该事务中的候选项集来减少计算支持度的时间。

修改后的事务数据库用 $\overline{C}_k$ 表示，在之前的若干次遍历中，$\overline{C}_k$ 比较大，效率低；当 $\overline{C}_k$ 能够放在内存中的时候，效率较高。因此，算法 AprioriHybrid 合并了 Apriori 和 AprioriTid，在之前的遍历中使用 Apriori，当 $\overline{C}_k$ 能够放在内存的时候，转换到 AprioriTid。所以，AprioriHybrid 的效率比 Apriori 和 ArioriTid 都要高。

2．深度优先算法

深度优先算法不产生候选项集，而是利用模式增长的方法。典型的深度优先算法包括 FP-growth、Eclat、H-Mine 等算法。

针对 Apriori 算法的固有缺陷（需要多次遍历数据库），J.Han 等提出了不产生候选频繁项集的方法：FP-growth 算法。采用分而治之的策略，在经过第一遍扫描之后，把数据库中的频繁项集压缩进一棵频繁模式树（FP-tree），同时依然保留其中的关联信息，随后再将 FP-tree 分化成一些条件库，然后再对这些条件库分别进行挖掘。实验表明，FP-growth 对不同长度的规则都有很好的适应性，同时在效率上相比 Apriori 算法有巨大的提高。

研究表明，没有哪种算法是绝对优秀的，它们仅在特定的支持度下执行效率比其他算法高。支持度越小，挖掘出来的频繁项集和关联规则越多，频繁项集的最大长度越长。广度优先算法在挖掘短的频繁项集时效率较高，深度优先算法在挖掘长的频繁项集时效率较高。

由于许多应用问题远比购物篮问题更复杂，大量研究从不同的角度对关联规则挖掘算法做了扩展，将更多的因素集成到关联规则挖掘方法之中，以此丰富关联规则的应用领域，拓宽支持管理决策的范围，如考虑属性之间的类别层次关系、时态关系、多表挖掘等。近年来围绕关联规则的研究主要集中于两个方面，即扩展经典关联规则能够解决问题的范围，提高经典关联规则挖掘算法的效率和规则的有效性。

5.2 Apriori 算法

5.2.1 Apriori 算法原理

Apriori 算法是发现关联规则领域的经典算法。该算法将发现关联规则的过程分为两个阶段：第一阶段，通过迭代，检索出事务数据库中的所有频繁项集，即支持度不低于用户设定的阈值的项集；第二阶段，利用频繁项集构造出满足用户最小置信度（或者其他设定的要求）的规则。

具体做法是：首先找出所有的频繁 1-项集，记为 L_1；然后利用 L_1 来产生候选 2-项集组成的集合 C_2，对 C_2 中的 2-项集进行判定挖掘出频繁 2-项集组成的集合 L_2；不断如此循环下去直到无法发现更多的频繁 k-项集为止。每挖掘一层 L_k 就需要扫描一遍整个数据库。

为提高频繁项集逐层产生的效率，一种称做 Apriori 性质的重要性质用于压缩搜索空间。

Apriori 性质：频繁项集的所有非空子集都必须也是频繁的。

Apriori 性质基于如下观察：根据定义，如果项集 I 不满足最小支持度阈值 min_sup，则 I 是非频繁的，即 $P(I) < \text{min_sup}$。如果项 A 添加到 I，则结果项集（即 $I \cup A$）不可能比 I 更频繁出现。因此，$I \cup A$ 也不是频繁的，即 $P(I \cup A) < \text{min_sup}$。

1. 生成频繁项集

Apriori 生成频繁项集的具体步骤如下：

（1）根据频繁$(k-1)$-项集组成的集合 L_{k-1}，产生全部候选 k-项集 C_k：令 p 和 q 是 L_{k-1} 中任意两个不同的项集（p 和 q 中的项按照字典次序大小排列），如果 p 的前（$k-2$）个项和 q 的前（$k-2$）个项都相同，并且 q 的最后一个项比 p 的最后一个项要大，那么把 q 的最后那个项加到 p 的后面，使之成为一个候选的 k-项集。依此找出所有的候选的 k-项集，组成 C_k。

（2）对 C_k 进行修剪：对 C_k 中的每一个项集 w，检查 w 的每一个（$k-1$）子集是否为频繁项集，只要有一个子集不是频繁项集，就将 w 从 C_k 中删除。

（3）计算 C_k 中每一个项集 w 的支持度：$\text{support} = \dfrac{N_i}{N}$，其中 N_i 是包含项集 w 的事务的数量，N 是全部事务的数量。

（4）将 support≥min_sup 的项集添加到频繁 k-项集 L_k 中。

（5）只要能够找到频繁 k-项集，并且 k 小于用户预先定义的最大值 k_{max}，重复上面的步骤，寻找频繁$(k+1)$-项集。

下面以表 5-1 中的数据为事务数据库，用上面阐述的步骤，挖掘其中的频繁项集。

取最小支持度阈值 min_sup=0.22。

候选 1-项集为 C_1={{牛奶}，{果冻}，{啤酒}，{面包}，{花生酱}}。计算各 1-项集的支持度为：

{牛奶}：support=6/9=0.67≥min_sup；　{果冻}：support=7/9=0.78≥min_sup；
{啤酒}：support=6/9=0.67≥min_sup；　{面包}：support=2/9=0.22≥min_sup；
{花生酱}：support=2/9=0.22≥min_sup。

因此，频繁 1-项集为 L_1={{牛奶}，{果冻}，{啤酒}，{面包}，{花生酱}}。

然后根据 L_1 生成候选 2-项集 C_2：

C_2={{牛奶，果冻}，{牛奶，啤酒}，{牛奶，面包}，{牛奶，花生酱}，{果冻，啤酒}，{果冻，面包}，{果冻，花生酱}，{啤酒，面包}，{啤酒，花生酱}，{面包，花生酱}}。由于 C_2 中各项集的子集均为频繁 1-项集，因此没有项集被修剪。计算各 2-项集的支持度为：

{牛奶，果冻}：support=4/9=0.44≥min_sup；{牛奶，啤酒}：support=4/9=0.44≥min_sup；
{牛奶，面包}：support=1/9=0.11<min_sup；{牛奶，花生酱}：support=2/9=0.22≥min_sup；
{果冻，啤酒}：support=4/9=0.44≥min_sup；{果冻，面包}：support=2/9=0.22≥min_sup；
{果冻，花生酱}：support=2/9=0.22≥min_sup；{啤酒，面包}：support=0<min_sup；
{啤酒，花生酱}：support=1/9=0.11<min_sup；{面包，花生酱}：support=0<min_sup。

因此，频繁 2-项集为 L_2={{牛奶，果冻}，{牛奶，啤酒}，{牛奶，花生酱}，{果冻，啤酒}，{果冻，面包}，{果冻，花生酱}}。

然后根据 L_2 生成候选 3-项集 C_3：

C_3={{牛奶，果冻，啤酒}，{牛奶，果冻，花生酱}，{牛奶，啤酒，花生酱}，{果冻，啤酒，面包}，{果冻，啤酒，花生酱}，{果冻，面包，花生酱}}。

对 C_3 进行修剪：

{牛奶，啤酒，花生酱}的子集{啤酒，花生酱}不是 L_2 的元素，即不是频繁项集，从 C_3 删除；

{果冻，啤酒，面包}的子集{啤酒，面包}不是频繁项集，从 C_3 删除；

{果冻，啤酒，花生酱}的子集{啤酒，花生酱}不是频繁项集，从 C_3 删除；

{果冻，面包，花生酱}的子集{面包，花生酱}不是频繁项集，从 C_3 删除。

计算各 3-项集的支持度为：

{牛奶，果冻，啤酒}：support=2/9=0.22≥min_sup；

{牛奶，果冻，花生酱}：support=2/9=0.22≥min_sup。

因此，频繁 3-项集为 L_3={{牛奶，果冻，啤酒}，{牛奶，果冻，花生酱}}。

然后根据 L_3 生成候选 4-项集 C_4：C_4={{牛奶，果冻，啤酒，花生酱}}，但由于{牛奶，果冻，啤酒，花生酱}的子集{果冻，啤酒，花生酱}不在 L_3 中，所以将其删除。

所以，本例的全部频繁项集为：$L = L_2 \cup L_3$，频繁 1-项集被排除在外，因为一个项无法构成关联规则。

2．根据频繁项集产生关联规则

在得到全部频繁项集 L 后，算法根据这些频繁项集产生关联规则，步骤如下：

（1）对于 L 中的每一个频繁项集 l，产生 l 的所有非空子集。

（2）对于 l 的每个非空子集 A，如果满足设定的评估准则（默认的评估准则是规则的置信度大于等于最小置信度阈值，即 $\frac{\text{support}(l)}{\text{support}(A)} \geqslant \text{min_conf}$，support($l$)和 support($A$)分别是项集 l 和 A 的支持度)，则输出规则“$A \Rightarrow \overline{A}$”，其中 $\overline{A} = l - A$。

一个生成的规则是否最终被保留下来，要看它是否满足评估准则。虽然默认的评估准则是置信度大于等于最小置信度阈值，Apriori 算法同时提供了多种对规则进行评估的方法，以从不同的侧面来考察一个规则，并最终确定那些真正让我们感兴趣的规则。这些评估准则都是基于两个基本的参数，即“前置信度”（Prior Confidence）和“后置信度”（Posterior Confidence），它们的定义如下：

前置信度 $C_{\text{prior}} = \frac{c}{N}$，后置信度 $C_{\text{posterior}} = \frac{r}{a}$。其中，$c$ 是规则后项的支持度，a 是规则前项的支持度，r 是规则前项和后项的联合支持度（即“前项∪后项”组成的项集的支持度，也称“规则支持度”），N 是数据库中的事务总数。

对于规则的度量值 e，Apriori 算法提供以下几种对规则的评估度量：

（1）规则的置信度。默认的评估度量，就是直接用后置信度来表示：

$$e = C_{\text{posterior}}$$

（2）置信度差。用前、后置信度的差的绝对值来表示：

$$e = |C_{\text{posterior}} - C_{\text{prior}}|$$

在结果不呈均匀分布时，该选项能避免偏倚。这有助于避免“明显”的规则被保留在规则集中。比如，实际情况可能是有80%的顾客购买最畅销的产品。预测85%的顾客购买该产品的规则并不能增加多少知识，尽管从绝对数的角度看85%的精确度非常好。

（3）置信度比率。这种度量基于前、后置信度的比值：

$$e = 1 - \min\left(\frac{C_{\text{posterior}}}{C_{\text{prior}}}, \frac{C_{\text{prior}}}{C_{\text{posterior}}}\right)$$

该度量尤其擅长发现预测罕见事件的规则。比如，假定有一种罕见的病情仅在1%的病人中发生，能够在10%的时间里预测这种情况的规则相对于随机猜测而言是很大的进步，但是从绝对数的角度看，10%的精确度似乎并不具有很强的说服力。

（4）信息差。这是一种基于信息增益的度量方法：

$$e = \frac{r\log\left(\frac{r}{ac}\right) + (a-r)\log\left(\frac{a-r}{a\bar{c}}\right) + (c-r)\log\left(\frac{c-r}{\bar{a}c}\right) + (1-a-c+r)\log\left(\frac{1-a-c+r}{\bar{a}\,\bar{c}}\right)}{\log(2)}$$

其中，r是规则的支持度，a是前项支持度，c是后项支持度，$\bar{a} = 1-a$，$\bar{c} = 1-c$。

信息差是给定前提条件后的信息增益与只给定结论先验置信度时的信息增益之间的差值。这种度量的一个重要特征是考虑了支持度，这样在给定置信水平下，覆盖记录更多的规则易被保留。

（5）标准化卡方（Normalized Chi-square）。这种度量方法是基于对独立的离散型数据的卡方统计检验：

$$e = \frac{(ac-r)^2}{a\bar{a}c\bar{c}}$$

这项度量是前提条件与结论之间相关性的统计指数。这项度量经过标准化，取值为0～1。这项度量比信息差更依赖于支持度。

以前文中生成的频繁项集为例，例如{牛奶，果冻，花生酱}，以规则的置信度为评估度量，设最小置信度阈值min_conf=70%，产生强规则的过程如下：

{牛奶，果冻，花生酱}的所有非空子集有：{牛奶，果冻}，{牛奶，花生酱}，{果冻，花生酱}，{牛奶}，{果冻}，{花生酱}，列出所有的规则及其置信度为：

$R1$：牛奶∧果冻⇒花生酱，$\text{confidence} = \frac{0.22}{0.44} = 50\% < \text{min_conf}$；

$R2$：牛奶∧花生酱⇒果冻，$\text{confidence} = \frac{0.22}{0.22} = 100\% \geqslant \text{min_conf}$；

$R3$：果冻∧花生酱⇒牛奶，$\text{confidence} = \frac{0.22}{0.22} = 100\% \geqslant \text{min_conf}$；

$R4$：牛奶⇒果冻∧花生酱，$\text{confidence} = \frac{0.22}{0.67} = 33\% < \text{min_conf}$；

$R5$：果冻⇒牛奶∧花生酱，$\text{confidence} = \frac{0.22}{0.78} = 28\% < \text{min_conf}$；

$R6$：花生酱 ⇒ 牛奶 ∧ 果冻， $\text{confidence} = \dfrac{0.22}{0.22} = 100\% \geqslant \text{min_conf}$。

因此，根据频繁项集{牛奶，果冻，花生酱}产生的强规则是 $R2$、$R3$ 和 $R6$。

3．Apriori 算法描述

整个算法由两个部分组成：Apriori 算法和 Apriori-gen 算法。其中 Apriori 算法为主算法，其中调用了 Apriori-gen 算法，用以根据频繁$(i-1)$-项集 L_{i-1} 生成候选 i-项集 C_i。

（1）Apriori 算法。

输入：

```
    I      //基本项目集合
    D      //事务数据库
    s      //支持度阈值
```

输出：

```
    L      //频繁项集
```

算法：

```
    k=0;   //k 表示遍历数据库的次数
    L=∅;
    C1= I;     //候选 1-项集为基本项目集合
    Repeat
        k=k+1;
        Lk=∅;
        for each   Ii∈Ck   do
            { c=0;     //每一个项集的初始计数设为 0
             for each t∈D do
                  if Ii∈t then c=c+1;}
      if c>=(s×|D|)    then Lk=Lk∪Ii;
      L=L∪Lk;
      Ck+1= Apriori-Gen (Lk)
    Until Ck+1=∅
```

（2）Apriori-gen 算法。

输入：

```
    Li–1       //频繁(i–1)-项集
```

输出：

```
    Ci        //候选 i-项集
```

算法：

```
    Ci =∅;
    for each I∈Li–1 do
        for each J≠I∈Li–1 do
            if i–2 of the elements in I and J are equal then
```

$$C_k = C_k \cup \{I \cup J\};$$

5.2.2　在 Clementine 中应用 Apriori 算法

执行关联规则挖掘的主要难点在于，有太多的可能的规则。就拿购物篮分析来说，一个超市的产品种类成千上万，挖掘后生成的规则数量也将十分庞大。很显然，这么多的规则，不可能一个一个地进行检查和处理。必须要有有效的算法，来压缩搜索空间，降低计算代价，但同时又不会丢失重要的规则。

在 Clementine 中执行 Apriori 算法的方法是由 Christian Borgelt 和 Rudolf Kruse 提出的。为了找到一个频繁项集，就必须计算包含这个频繁项集的事务数量。在 Christian Borgelt 和 Rudolf Kruse 提出的方法中，把这些项集的计数器组织起来，成为一种特殊的"前缀树"（Prefix Tree），树上的每一个节点代表了一个项集的计数器。这种数据结构可以以很小的内存需求来有效地存储计数信息，同时也可以用于对事务的处理，以生成规则。关于该方法的细节，可以参见相关的资料（http://fuzzy.cs.uni-magdeburg.de/ ~borgelt/doc/apriori/apriori.html）。

本例在 Clementine 中应用 Apriori 节点来对某超市的客户采购数据集进行购物篮分析。该数据集包含有 22 个属性（这些属性包括：COD、pasta、milk、water、biscuits、coffee、brioches、yoghurt、frozen vegetables、tunny、beer、tomato、souce、coke、rice、juices、crackers、oil、frozen fish、ice cream、mozzarella、tinned meat。其中 COD 是记录编号，其他 20 个属性代表 20 种商品），共 46243 个记录。每个属性代表某种商品，其取值为"0"或者"1"，"0"表示没有购买该商品，"1"表示购买了该商品。数据集存储在 Transactions.txt 文件中，如表 5-2 所示。

表 5-2　购物篮分析数据集

COD	pasta	milk	water	biscuits	coffee	brioches	yoghurt	…	tinned meat
1	0	0	0	0	0	0	0	…	1
2	0	1	0	0	0	0	0	…	0
3	1	1	0	0	0	0	0	…	0
4	0	0	1	0	0	0	0	…	0
5	0	0	0	0	0	0	0	…	0
6	1	1	0	1	0	0	0	…	0
7	0	1	0	1	0	0	0	…	0
8	1	0	0	0	0	0	0	…	0
9	0	0	0	0	0	0	0	…	0
10	0	0	0	1	0	0	0	…	0
11	0	0	0	0	0	0	0	…	1
12	1	1	0	0	1	0	0	…	0
…	…	…	…	…	…	…	…	…	…
46243	0	0	1	0	0	0	0	…	0

对该数据集用 Apriori 节点进行购物篮分析的数据流如图 5.1 所示。

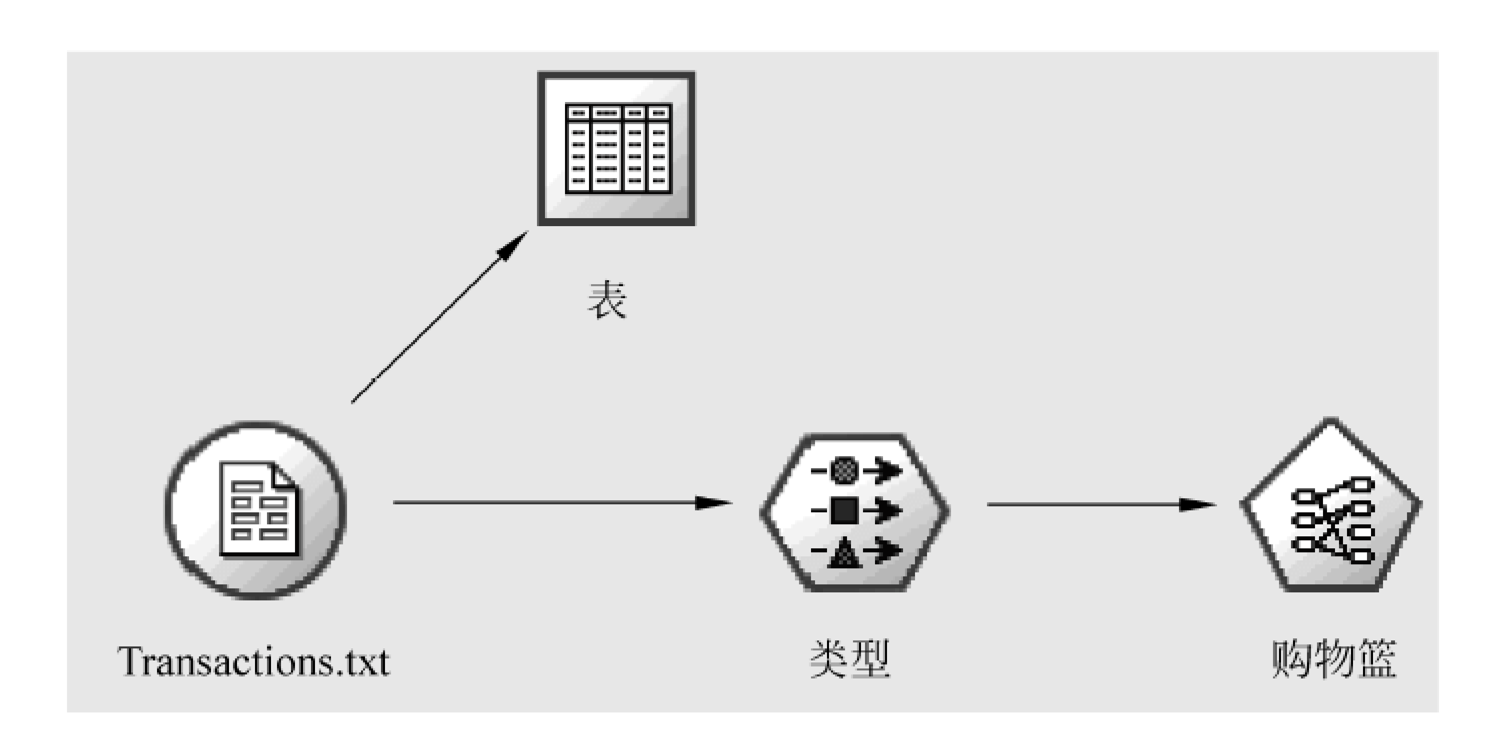


图 5.1　购物篮分析数据流

1. 生成模型

首先，将“数据源”中的“可变文件”节点添加到数据流区域，并将 Transactions.txt 文件加载到该节点，如图 5.2 所示。



图 5.2　设置数据源节点

注意，由于在 Transactions.txt 文件中，属性之间是用制表符来间隔的，所以在“定界符”选项中选中“选项卡”(选项卡，即 Tab，也就是制表符)。

向数据流中添加“表”节点，并建立从 Transactions.txt 节点到“表”节点的连接，

执行“表”节点，即可浏览数据集中的数据，如图 5.3 所示。

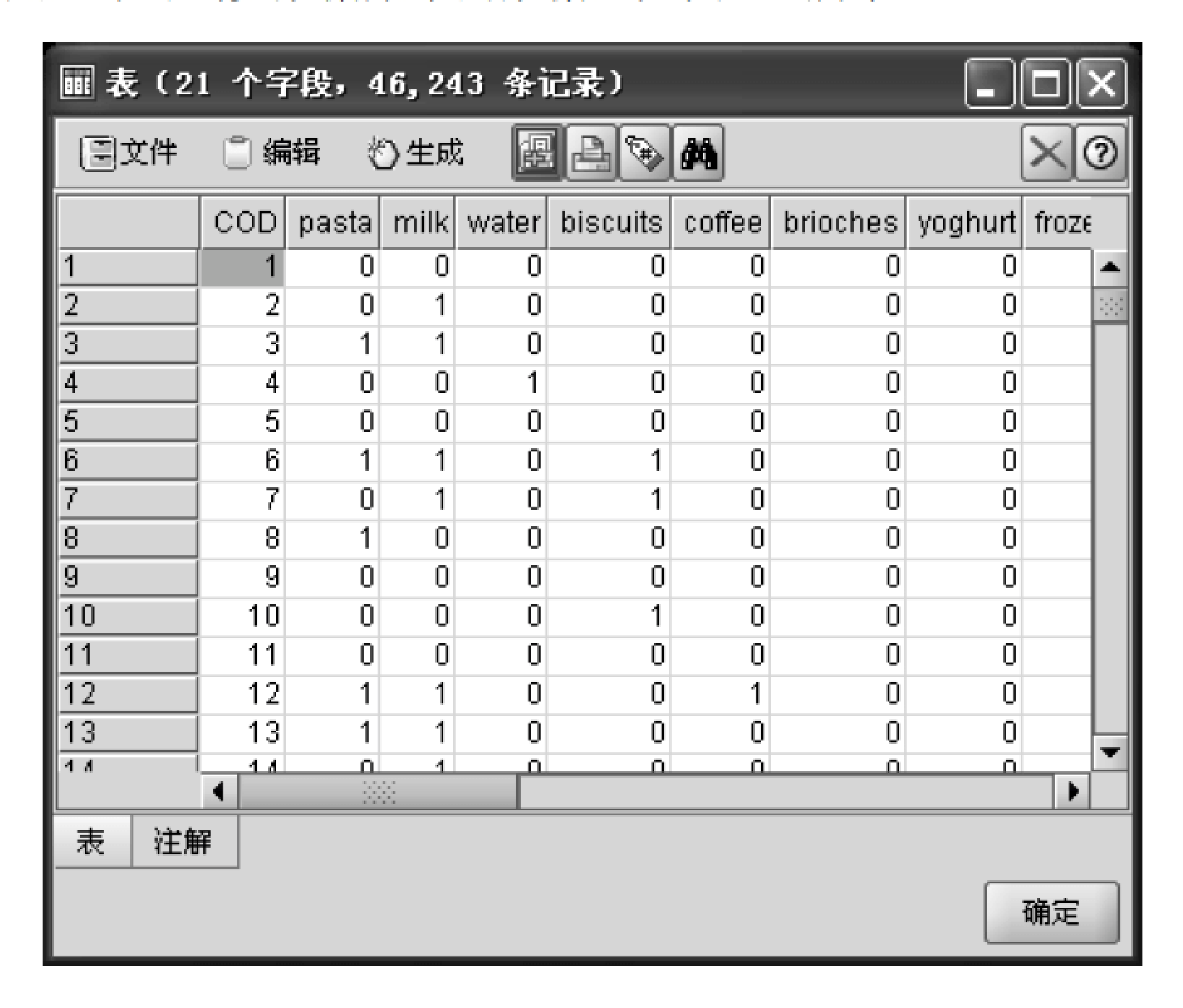


图 5.3 浏览数据集

向数据流中添加“类型”节点，并建立从 Transactions.txt 节点到“类型”节点的连接，对“类型”节点进行设置：将 COD 字段的方向设置为“无”（因为该字段只是记录编号，没有实际意义，因此不参与建模），把其他 20 个字段的类型设置为“标志”，并把它们的方向都设置为“两者”（即输入和输出双向），最后单击“读取值”按钮，如图 5.4 所示。



图 5.4 设置“类型”节点

向数据流中添加 Apriori 节点，建立从“类型”节点到 Apriori 节点的连接，然后对

Apriori 节点进行设置，在“模型”标签下的设置如图 5.5 所示。



图 5.5　设置 Apriori 节点

图 5.5 中的各选项说明如下：

模型名称（Model Name）：指定要产生的模型名称。

- 自动（Auto）：选择该选项后，模型名称将根据目标字段或者后项字段自动生成。这是默认的设置。
- 自定义（Custom）：选择该选项可以为节点创建的模型指定用户定义的模型名称。

这里选择“自定义”，为模型取名为“购物篮”。

使用分区数据（Use Partitioned Data）：如果用户定义了分割数据集，选择训练数据集作为建模数据集，并利用测试数据集对模型评价。

最低条件支持度（Minimum Antecedent Support）：用户自定义最小支持度阈值。注意，这个支持度是指规则的前项支持度，即规则的 IF 部分的支持度，而不是指规则支持度。如果得到的规则适用于数据集的很小一部分子集，尝试提高该项设置。这里将最低条件支持度设置为“5.0”。

最小规则置信度（Minimum Rule Confidence）：用户自定义最小置信度阈值。置信度比该阈值小的规则将被丢弃。如果得到的规则太多，尝试提高该项设置，如果得到的规则太少（或者根本就没有规则），尝试降低该项设置。这里将最小规则置信度设置为“50.0”。

最大前项数（Maximum Number of Antecedents）：可以指定任一规则的最大前项数。这是限制规则复杂程度的一种方法。如果规则过于复杂或者过于具体，尝试降低该项设置。该项设置对训练时间也有很大影响。如果对规则集的训练时间过长，尝试降低该项设置。这里将最大前项数设置为“5”。

仅包含标志变量的真值（Only True Values for Flags）：如果选择了该选项，只有真值会出现在最终的规则中。这有助于理解规则。

优化（Optimize）：选择“速度”（Speed）使算法运行加快，但是占用更多的内存。选择“内存”（Memory）使运行速度减慢，但是节约内存。

在 Apriori 节点的“专家”标签下可以对节点进行高级选项设置，如图 5.6 所示。



图 5.6　Apriori 节点的高级选项

将“模式”选项设置为“专家”，可以选择不同的评估尺度。Apriori 提供 5 种评估潜在规则的方法。这里选择“规则置信度”。

允许没有前项的规则（Allow Rules Without Antecedents）：选中该选项允许规则只有后项而没有前项。当用户关心哪些项是频繁发生的时候，可以选择该选项。

例如，“罐装蔬菜”是一个没有前项的 1-项集规则，它表示购买罐装蔬菜在数据集中是频繁发生的。在很多情况下，用户可能要做出最确信的预测，因此需要这些没有前项的规则。该选项默认是未选中的。按照约定，没有前项的规则的“前项支持度”为 100%，因此规则的支持度就等于置信度。

对 Apriori 节点设置完毕后，单击“执行”按钮，即可在管理器窗口的“模型”标签下生成“购物篮”模型。

2．浏览模型

对生成的模型进行浏览，可以看到所有的规则及其相关信息。右击“模型”标签下生成的“购物篮”模型，在快捷菜单中选择“浏览”命令，即可打开模型对话框，如图 5.7 所示。

在对话框的“模型”标签下，可以看到一张表，显示了 Apriori 算法根据数据集生成的规则。表中的每一行代表了一个规则。第一列代表规则后项（即规则的 then 部分），第二列代表规则前项（即规则的 If 部分）。后面的几列显示了规则的相关信息，如置信度、支持度等，如果想显示规则的其他信息，可以单击对话框上方的“显示/隐藏标准”按钮。下列 7 项信息可以在表中显示：

（1）**规则 ID（Rule ID）**：显示规则的编号。

（2）**实例（Instances）**：显示数据集中包含规则前项的事务数量。例如规则“biscuits and water ⇒ milk”的实例为“2734”，表示在数据集中有 2734 个事务包含规则的前项{biscuits, water}。

购物篮

文件 生成

按以下内容进行排序: 置信度 % 14 属于 14

后项	前项	支持度 %	置信度 %
milk	biscuits water	5.912	60.241
milk	yoghurt pasta	5.279	58.91
milk	biscuits pasta	7.763	57.855
milk	brioches pasta	5.949	57.361
pasta	tomato_souce milk	5.947	57.236
milk	coffee pasta	5.994	56.926
milk	water pasta	9.552	55.988
milk	tomato_souce pasta	6.174	55.131
milk	juices	8.256	53.274
pasta	tomato_souce	11.597	53.235
milk	yoghurt	15.235	52.165
pasta	rice	6.503	51.812
milk	biscuits	20.474	51.531
milk	tomato_souce	11.597	51.277

模型 汇总 注解

确定

图 5.7 浏览生成的关联规则模型

（3）**支持度（Support）**：显示规则的前项支持度。它是“包含规则前项的事务数量”（即实例）占全部事务数量的比例。例如“biscuits and water $\Rightarrow$ milk”的支持度为：$\frac{2734}{46243}\times 100\% = 5.912\%$。

（4）**规则支持度（Rule Support）**：显示“同时包含规则前项和后项的事务数量”占全部事务数量的比例。例如规则“biscuits and water $\Rightarrow$ milk”的规则支持度为 3.562%，表示数据集中有 3.562%的事务包含项集{biscuits, water, milk}。

（5）**置信度（Confidence）**：显示规则支持度和前项支持度的比值。例如规则“biscuits and water $\Rightarrow$ milk”的置信度为：$\frac{3.562\%}{5.912\%}\times 100\% = 60.241\%$。

（6）**提升（Lift）**：用规则的置信度除以规则后项的支持度所得的比值，即为提升。例如规则“biscuits and water $\Rightarrow$ milk”的置信度为 60.241%，其后项“milk”的支持度为 46.132%（在全部 46243 个事务中，有 21333 个事务包含“milk”，因此“milk”的支持度为 $\frac{21333}{46243}\times 100\% = 46.132\%$），所以该规则的提升 $\text{Lift} = \frac{60.241\%}{46.132\%} = 1.306$。它的具体含义是这样的：在 46243 个客户组成的人群中，购买“milk”的概率为 46.132%，如果

对人群加以限制，即购买了“biscuits”和“water”的客户组成的人群，那么这个人群购买“milk”的概率为 60.241%，通过对人群进行特定的限制，可以使“milk”被购买的概率提高 1.306 倍。如果某个规则的提升值接近于 1，就说明这个规则前项的条件限制并没有使这个规则后项发生的概率提高多少。所以，提升值比 1 大很多的那些规则比提升值接近于 1 的规则更有价值。

（7）**部署能力（Deployability）**：它是一个比值，即那些支持规则前项但不支持规则后项的事务，占全部事务的比例。例如规则“biscuits and water $\Rightarrow$ milk”，支持“biscuits and water”但不支持“milk”的事务数量为 1087，所以该规则的部署能力为：$\text{Deployability} = \frac{1087}{46243} \times 100\% = 2.351\%$，表示有 2.351%的客户虽然购买了“biscuits”和“water”，但却并没有购买“milk”。因此，一个部署能力比较大的规则，说明该规则后项的发生概率还有较大的提高空间。

另外，用户可以通过单击“排序方式”后面的下拉列表框选择多种排序方式。有 6 种排序方式：支持度、置信度、规则支持度百分比、后项、提升、部署能力。通过单击▲按钮，可实现“升序”和“降序”的切换。

当生成的规则很多，而只想显示某些特定的规则时，可以单击“显示过滤器”按钮，并打开“编辑过滤器”对话框，设定规则过滤规则，然后启用过滤器，即可在规则列表中只显示经过滤后的规则。例如，如果只想显示后项包含“milk”、前项包含“pasta”的规则，则可以按照如图 5.8 所示来设置过滤器，然后单击“确定”按钮即可。



图 5.8 编辑规则过滤器

在对话框的“汇总”标签中，则显示了该模型的一些概要信息，例如规则数、最大支持度、最大提升等，也显示了关于本次建模的其他记录，如图 5.9 所示。

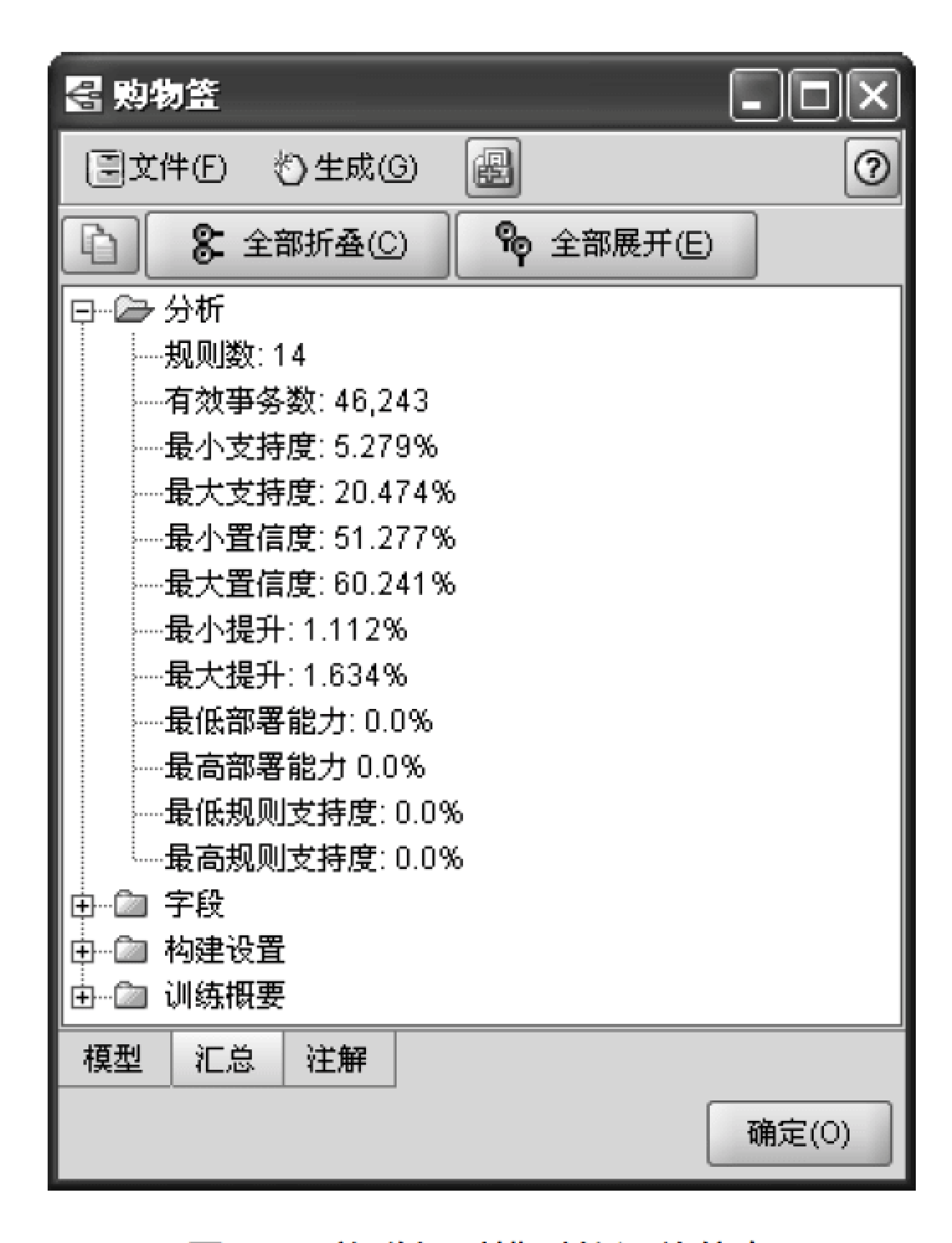


图 5.9　关联规则模型的汇总信息

5.3 CARMA 算法

5.3.1 CARMA 算法原理

CARMA（the Continuous Association Rule Mining Algorithm）是一种比较新的关联规则算法，它是 1999 年由 Berkeley 大学的 Christian Hidber 教授提出来的。它是一种占用内存少，能够处理在线连续交易流数据的一种新型的关联规则挖掘算法。它有以下特点：算法执行过程中即能不断产生部分计算结果，供用户参考；在算法执行过程中，用户能根据产生的部分计算结果控制算法如何进行下去；算法给出的结果必须是精确的。在线挖掘关联规则的算法允许用户随时调整最小支持度阈值，以得出合理的结果，如果中间结果已经令人满意，用户也可以随时终止算法的执行。

和 Apriori 算法一样，CARMA 算法也包括两个阶段。第一阶段根据事务数据库产生频繁项集，第二阶段根据频繁项集产生关联规则，其中第一阶段的处理是算法的核心部分。

为了发现频繁项集，CARMA 需要遍历整个事务数据库两次，分别由两个算法来实现。第一次遍历称为 PhaseI，第二次遍历称为 PhaseII。PhaseI 产生一个频繁项集的候选集，称为潜在的频繁项集的集合。PhaseII 把 PhaseI 产生的集合进行删减，得出最终的结果。由于算法本身的特点，第二次遍历并不一定需要进行完，甚至很多时候第二阶段不需要做。另外，CARMA 算法还有一个很好的性质，就是在遍历事务数据库的过程中可

以不断地改变支持度阈值，以控制输出规则的大小和数目，这是其他关联规则挖掘算法所没有的。

1. PhaseI 算法

在第一次遍历中，PhaseI 生成一个由所有潜在的频繁项集组成的超集 V，并且用一个栅格数据结构（lattice）来存储 V 中每个项集的相关信息。在遍历数据库的过程中，PhaseI 按照事务编号一个一个地处理数据库中的事务，在每一个事务处理之后，会在栅格里添加或者删除一些项集。对于每一个项集 v，PhaseI 会存储下面 3 个参数：

Count(v)：自从 v 被添加到 V 中后，包含项集 v 的事务的数量。

firstTrans(v)：当 v 被添加到栅格时的当前事务号。

maxMissed(v)：v 在添加到栅格之前，包含项集 v 的事务的数量。

栅格的结构如表 5-3 所示。

表 5-3　栅格的结构示例

项集	count	firstTrans	maxMissed
i_1	count(i_1)	firstTrans (i_1)	maxMissed (i_1)
i_2	count(i_2)	firstTrans (i_2)	maxMissed (i_2)
…	…	…	…
i_k	count(i_k)	firstTrans (i_k)	maxMissed (i_k)

图 5.10 说明了对于某个项集{a,b}以上 3 个参数所代表的含义。

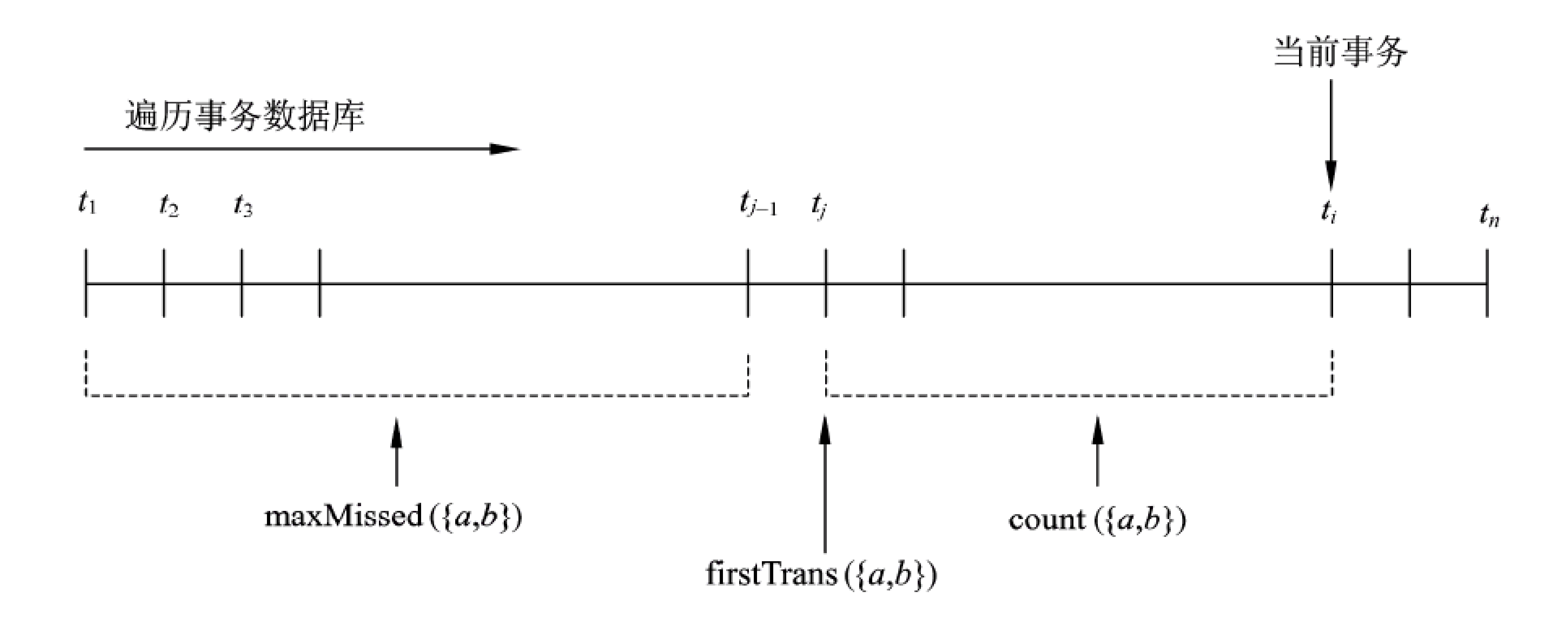


图 5.10　3 个参数的含义

设 t_1, t_2, ⋯, t_n 是事务数据库中的 n 个事务，项集{a,b}在遍历到事务 t_j 时被添加到 V 中，当前读取的事务是 t_i。那么 maxMissed({a,b})是 t_1～t_{j-1} 的 j–1 个事务中包含项集{a,b}的事务的个数。firstTrans({a,b})就等于 j。count({a,b})是 t_j～t_i 的（i–j+1）个事务中包含项集{a,b}的事务的个数。

假设现在读取事务 i，而且已经有了一个上面这样的栅格。对于栅格中的任何项集 v，可以得到前 i 个事务对项集 v 的支持度的下限值 count(v)/i 和上限值[maxMissed(v)+count(v)]/i。分别称它们为最小支持度 minSupport(v)和最大支持度 maxSupport(v)，即：

最小支持度 minSupport(v)=count(v)/i

最大支持度 maxSupport(v)=[maxMissed(v)+count(v)]/i

在遍历事务数据库的过程中，每处理一个事务，用户可以自由地设定一个支持度阈值。那么可以得到一个支持度序列 $\sigma=(\sigma_1,\sigma_2,\cdots)$，其中 σ_i 是处理第 i 个事务时设定的支持度阈值。用 $\text{avg}_i(\sigma)$ 来表示“到事务 i 为止，σ 的平均值”，即 $\text{avg}_i(\sigma)=\frac{1}{i}\sum_{j=1}^{i}\sigma_j$ 。

在 CARMA 中还定义了“到事务 i 为止，σ 的上限序列”，用 $\lceil\sigma_i\rceil$ 来表示，它是一个大于等于 σ 的支持度序列，包含的元素的数量和 σ 相同。

设 $\sigma=(\sigma_1,\sigma_2,\cdots,\sigma_n)$，当读取到第 i 个事务时：

（1）如果 $\sigma_i>\sigma_j$（j=1,2,…,i–1），即 σ_i 比之前的(i–1)个支持度都要大，那么 $\lceil\sigma_i\rceil$ 中的前 i 个元素均为 σ_i，其他(n–i)个元素均为 0，即 $\lceil\sigma_i\rceil=(\sigma_i,\sigma_i,\cdots,0,0)$ 。

（2）如果 σ_i 并不比之前的(i–1)个支持度都要大，则将其中比 σ_i 大的保留，比 σ_i 小的则用 σ_i 代替。

例如 $\sigma=(0.3,0.7,0.9,0.5)$，则：$\lceil\sigma_1\rceil=(0.3,0,0,0)$，$\lceil\sigma_2\rceil=(0.7,0.7,0,0)$，$\lceil\sigma_3\rceil=(0.9,0.9,0.9,0)$，$\lceil\sigma_4\rceil=(0.5,0.7,0.9,0.5)$ 。

下面阐述 PhaseI 产生 V 并更新栅格的具体过程：

一开始，PhaseI 令 V 仅包含一个元素，即空集，V={∅}，并设置其 count、firstTrans 和 maxMissed 为 0。注意，空集是任意非空项集的子集。

设算法已经处理了(i–1)个事务，现在读取第 i 个事务 t_i，它是一个项集。σ_i 是当前用户定义的支持度阈值，下面的过程包括 3 个步骤：

第一步：对于 V 中的每一个项集 v，如果 v 被包含在 t_i 中，则令 count(v)=count(v)+1。

第二步：对于 t_i 的每一个子集 v（包括 t_i 本身），如果 v 不在 V 中，那么当且仅当 v 的所有子集 w 已经被包含在 V 中且满足 maxSupport(w)≥σ_i 时，将 v 添加到 V 中，并令 count(v)=1，firstTrans(v)=i，然后计算 maxMissed(v)，这需要从两个方面来进行：

（1）因为 w 是 v 的子集，所以 maxSupport(w)≥maxSupport(v)。又因为：

maxSupport(w)=[maxMissed(w)+count(w)]/i，

maxSupport(v)=[maxMissed(v)+count(v)]/i，*count*(v)=1

所以有：maxMissed(w)+count(w)≥maxMissed(v)+1，即：

$$\text{maxMissed}(v)\leqslant\text{maxMissed}(w)+\text{count}(w)-1$$

（2）因为用户可以随时改变阈值，maxMissed(v)的计算还依赖于当前的和以前的支持度阈值。对于项集 v，前（i–1）个事务对项集 v 的支持度满足不等式：

$\text{support}_{i-1}(v)\leqslant\text{avg}_{i-1}(\lceil\sigma\rceil_{i-1})+\frac{|v|-1}{i-1}$，其中|$v$|表示 v 中包含的项的数量。

因为 maxMissed(v)=$\text{support}_{i-1}(v)\times(i-1)$，在不等式两边同乘以($i$–1)，同时由于 maxMissed($v$)是一个整数，所以有：

$$\text{maxMissed}(v)\leqslant\lfloor(i-1)\text{avg}_{i-1}(\lceil\sigma\rceil_{i-1})\rfloor+|v|-1$$

这里，$\lfloor x \rfloor$表示比实数 x 小的最大的整数。

结合以上两个方面，计算 maxMissed(v)为：

$$\min\{\lfloor (i-1)\mathrm{avg}_{i-1}(\lceil \sigma \rceil_{i-1}) \rfloor + |v| - 1, \max\mathrm{Missed}(w) + \mathrm{count}(w) - 1 \mid w \subset v\}$$

第三步：每处理完 k 个事务（k 默认设置为 500），检查 V 中的每一个项集，计算 maxSupport=(maxMissed+count)/i，i 是当前的事务号。如果 maxsupport $< \sigma_i$，就将该项集从 V 上删除（σ_i 是当前的支持度阈值）。

下面是一个根据事务序列生成 V 的例子。

设事务序列为 T=({a,b},{a,b,c},{b,c})，支持度阈值序列为 $\sigma = (0.3, 0.9, 0.5)$。首先初始化 V 为仅包含空集的集合 V={∅}。

（1）读取 t_1={a,b}，$\sigma_1 = 0.3$。

第一步，由于∅是 t_1 的子集（空集是任意非空项集的子集），所以将 V 中的 count(∅)加 1。

第二步，对于 t_1 的子集{a}，由于∅是{a}的子集，且：

maxSupport(∅)=[maxMissed(∅)+count(∅)]/i=(0+1)/1=1>0.3

所以将{a}添加到 V 中。同理，将{b}添加到 V 中。事实上，1-项集是自动添加到 V 中的。然后设置 count({a})=1，firstTrans({a})=1，count({b})=1，firstTrans({b})=1。由于 t_1 是第一个事务，所以 maxMissed(∅)、maxMissed({a})和 maxMissed({b})均为 0。因此处理 t_1 后生成的 V 如表 5-4 所示。

表 5-4　处理 t_1 之后的 V

项集	count	firstTrans	maxMissed	maxSupport
∅	1	1	0	1
{a}	1	1	0	1
{b}	1	1	0	1

（2）读取 t_2={a,b,c}，$\sigma_2 = 0.9$。

第一步，将∅、{a}、{b}的 count 加 1，因为它们均被包含在 t_2 中。

第二步，对于 t_2 的子集{c}，由于它是 1-项集，所以直接加入 V 中，且 count({c})=1，firstTrans({c})=2，maxMissed({c})=0。可以看出，1-项集都是在第一次出现时就被添加到 V 中的，因此它们的 maxMissed 总是为 0 的。

对于 t_2 的子集{a,b}，由于{a,b}的子集{a}和{b}都已经是 V 的元素且它们的 maxSupport 都是大于 $\sigma_2 = 0.9$ 的，所以将{a,b}添加到 V 中，并设置 count({a,b})=1，firstTrans({a,b})=2。

因为$\lceil \sigma_1 \rceil = (0.3, 0, 0)$，所以 $\mathrm{avg}_1(\lceil \sigma_1 \rceil) = 0.3$，且$\lfloor (2-1)\mathrm{avg}_1(\lceil \sigma_1 \rceil) \rfloor + 2 - 1 = 1$。同时，对于 w={a}和 w={b}，maxMissed(w)+count(w)=2。所以 maxMissed({a,b})=1。

而对于 t_2 的另外两个子集{a,c}和{b,c}，尽管它们的子集都已经是 V 的元素，但由于它们的子集{c}的 maxSupport({c})=1/2=0.5<$\sigma_2 = 0.9$，所以不能将它们添加到 V 中，t_2 本身{a,b,c}作为这两个子集的超集当然也是不频繁的。因此处理 t_2 后生成的 V 如表 5-5 所示。

表 5-5　处理 t_2 之后的 V

项集	count	firstTrans	maxMissed	maxSupport
$\varnothing$	2	1	0	1
$\{a\}$	2	1	0	1
$\{b\}$	2	1	0	1
$\{c\}$	1	2	0	0.5
$\{a,b\}$	1	2	1	1

（3）读取 $t_3=\{b,c\}$，$\sigma_3=0.5$。

第一步，将$\varnothing$、$\{b\}$、$\{c\}$的 count 加 1，因为它们均被包含在 t_3 中。

第二步，对于 t_3 本身$\{b,c\}$，由于$\{b,c\}$的子集$\{b\}$和$\{c\}$都已经是 V 的元素且：

$$\text{maxSupport}(\{b\})=3/3=1>\sigma_3=0.5，\text{maxSupport}(\{c\})=2/3=0.66>\sigma_3=0.5$$

所以，可以将$\{b,c\}$添加到 V 中，设置 count($\{b,c\}$)=1，firstTrans($\{b,c\}$)=3。

因为$\lceil\sigma_2\rceil=(0.9,0.9,0)$，所以 $\text{avg}_1(\lceil\sigma_1\rceil)=0.9$，且$\lfloor(3-1)\times0.9\rfloor+2-1=2$。同时，对于$\{b,c\}$的两个子集，maxMissed($\{b\}$)+count($\{b\}$)–1=2。所以 maxMissed($\{b,c\}$)=min$\{2,1\}$=1。因此处理 t_3 后生成的 V 如表 5-6 所示。

表 5-6　处理 t_3 之后的 V

项集	count	firstTrans	maxMissed	maxSupport
$\varnothing$	3	1	0	1
$\{a\}$	2	1	0	2/3=0.66
$\{b\}$	3	1	0	1
$\{c\}$	2	2	0	2/3=0.66
$\{a,b\}$	1	2	1	2/3=0.66
{b,c}	1	3	1	2/3=0.66

2．PhaseII 算法

在得到所有候选的频繁项集之后，由 PhaseII 算法来进行对事务数据集的第二次遍历，以确定最终的频繁项集。

PhaseII 算法的第一步是从 V 中删除那些不频繁的项集，它们的支持度小于支持度阈值序列$\sigma=(\sigma_1,\sigma_2,\cdots,\sigma_n)$中的最后一个阈值$\sigma_n$。

然后，遍历事务数据库。在处理一个事务 t_i 时，检查 V 中的每个项集 v 并更新栅格中的相关参数，步骤如下：

（1）如果 firstTrans(v)<i，v 被打上确定标记（表示确定为频繁项集）。如果 V 所有的项集均被打上确定标记，则无须再继续遍历数据库，PhaseII 算法即停止。

（2）如果 v 出现在当前事务 t_i 中，更新栅格中 v 的相关参数：

- ❑ count(v)= count(v)+1。
- ❑ maxMissed(v)= maxMissed(v)–1。
- ❑ 如果 firstTrans(v)=i，设置 maxMissed(v)=0。这一设置可能会使得 v 的某些超集 w 满足 maxSupport(w)>maxSupport(v)，如果 w 在 V 中，则调整它的 maxMissed(w)=

count(v) – count(w)。

- 如果 maxSupport(v)<σ_n，将 v 从 V 中删除。

和传统的关联规则算法（如 Apriori）相比，CARMA 算法是一种在线的关联规则算法，它可以以网络上不断产生的实时交易流为数据源，通过一遍扫描，最多两遍来构造满足给定支持度的规则集，而且 CARMA 算法允许在扫描过程中，用户可以对支持度进行修改。

在内存的使用上，CARMA 算法比其他关联规则算法更具优势，它在执行过程中需要的内存要比其他关联规则算法少很多，但是速度稍慢是 CARMA 算法的一个缺点。

5.3.2 在 Clementine 中应用 CARMA 算法

在 Clementine 中，CARMA 节点用来执行 CARMA 算法，并生成模型。

CARMA 节点对事务数据库的格式有所要求，它只能够处理以下两种格式的数据：

（1）事务格式（Transactional Format）

事务型数据的每个事务只包含一个项。如果一个顾客购买的商品不止一种，那么把每一种商品单独作为一个事务。

（2）表格式（Tabular data）

表格式中的项的取值是一个标志型数据，用“T”和“F”来表示。其中“T”（True）表示项出现在事务中，“F”（False）表示项没有出现在事务中。

图 5.11 中的（a）和（b）分别显示了用事务格式和表格式表示的一个数据库例子。

Customer	Purchase
1	jam
2	milk
3	jam
3	bread
4	jam
4	bread
4	milk

（a）

Customer	Jam	Bread	Milk
1	T	F	F
2	F	F	T
3	T	T	F
4	T	T	T

（b）

图 5.11 事务数据库的表示方法

本小节对某超市的交易数据进行关联分析，研究销售商品间的关联关系，目的是为超市货架的摆放提供科学的依据，对促销决策提供参考建议。

事务数据集为 Clementine 自带的 Baskets1n 数据集，存放在 Clementine 安装目录下的 Demos 文件夹中（...\clementine12.0\Demos\BASKETS1n），包含了 1000 个数据样本，每个样本包含了顾客的卡号、性别、年龄、收入、付款方式等一系列个人信息，以及其购买的各种食品清单，是一个 Tabular 格式的数据集。

完整的数据流如图 5.12 所示。

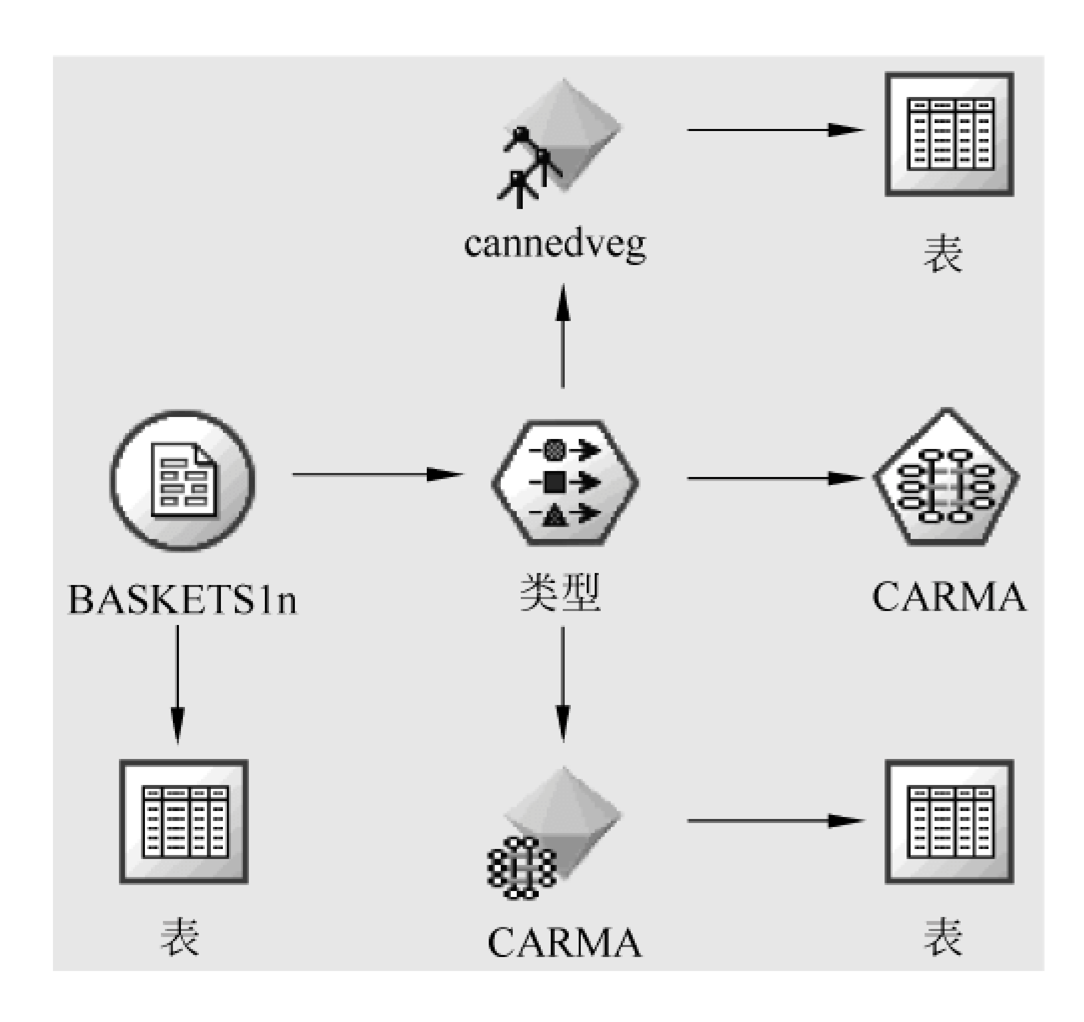


图 5.12　CARMA 分析数据流

1. 生成模型

首先，将“数据源”中的“可变文件”节点添加到数据流区域，并将 BASKETS1n 文件加载到该节点。

向数据流中添加“表”节点，并建立从数据源节点到“表”节点的连接，执行“表”节点，即可浏览数据集中的数据。

向数据流中添加“类型”节点，并建立从 BASKETS1n 节点到“类型”节点的连接。打开“类型”节点的编辑窗口，直接单击“读取值”按钮，可以自动选择各字段的数据类型，如图 5.13 所示。



图 5.13　设置“类型”节点

注意，这里无须指定各字段的“方向”，CARMA节点会自动选择所有的标志型字段，并把“方向”当作“两者”（即双向）来处理。非标志型字段会被CARMA节点自动忽略。

向数据流中添加建模节点CARMA节点，建立从“类型”节点到CARMA节点的连接，然后对CARMA节点进行设置，在节点编辑窗口的“字段”标签下，设置如图5.14所示。



图5.14　选择建模字段

图5.14中的各选项说明如下：

使用类型节点设置（Use Type Node Settings）：使用类型节点定义的所有标志型字段进行建模。

使用定制设置（Use Custom Settings）：用户自己选择需要的标志型字段进行建模。在本例中，有些标志型字段是不参与建模的，比如sex、homeown等，参与建模的字段只包括商品字段。所以，这里要选中“使用定制设置”单选按钮。

使用事务处理格式（Use Transactional Format）：用来为模型指定要处理的数据格式为transactional类型，系统默认格式为tabular。由于本例使用的数据集为tabular类型，因此不能选中该选项。

单击对话框右侧的按钮，打开“选择字段”对话框，从可用的标志型字段中选取参与建模的字段。可以按住Shift键不放并用鼠标选择连续的多个字段，也可以按住Ctrl键不放并选择单个的不连续的字段，然后单击“确定”按钮，如图5.15所示。

切换到编辑窗口的“模型”标签下，可对建模进行基本设置，如图5.16所示。



图 5.15　选择参与建模的字段

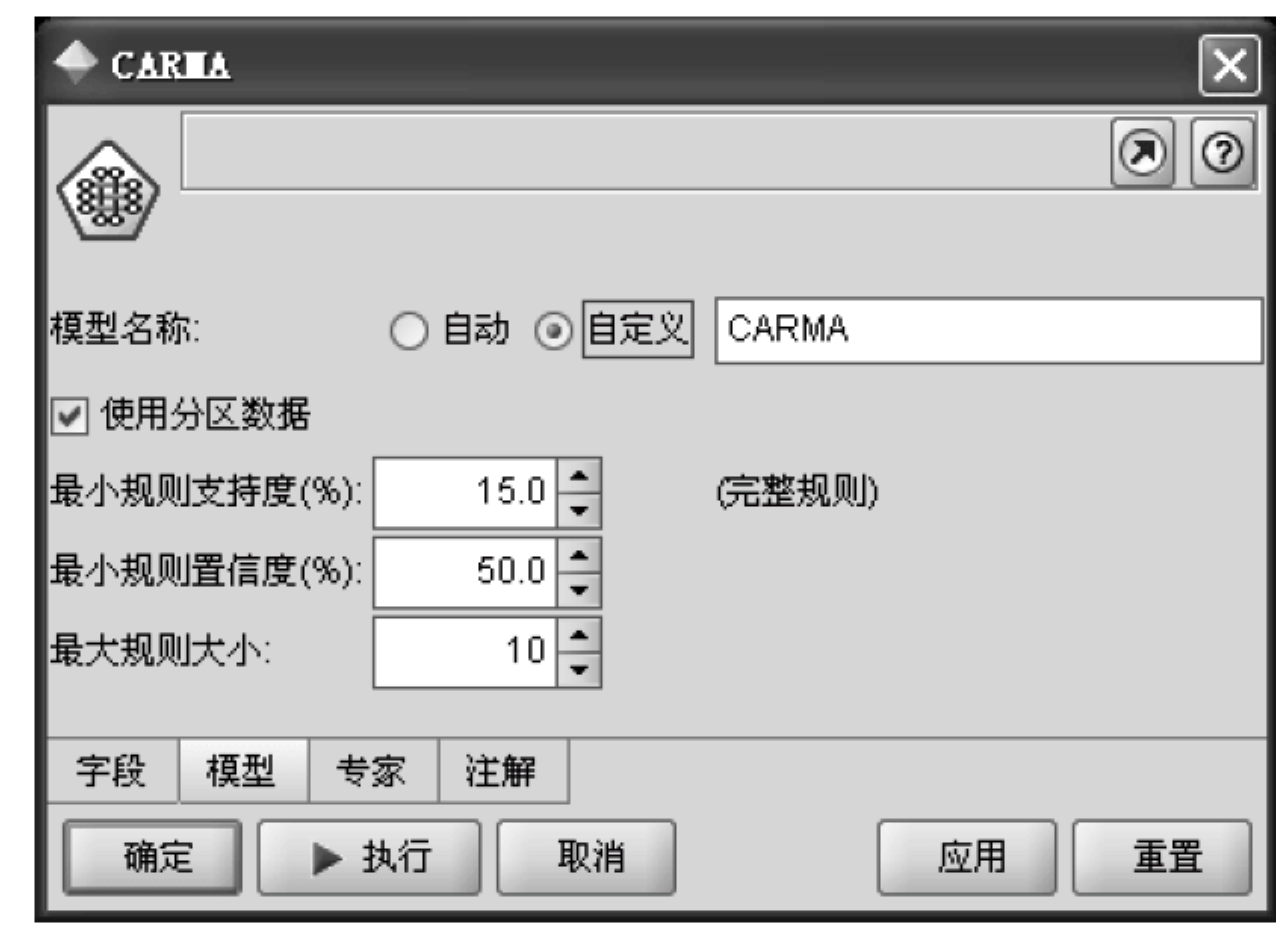


图 5.16　CARMA 模型基本设置

图 5.16 中的各选项说明如下：

模型名称（Model Name）：指定要产生的模型名称。

- ❑ 自动（Auto）：选择该选项后，模型名称将根据目标字段或者后项字段自动生成。这是默认的设置。
- ❑ 自定义（Custom）：选择该选项可以为节点创建的模型指定用户定义的模型名称。

这里选择“自定义”，为模型取名为“CARMA”。

最小规则支持度（Minimum Rule Support）：使用 CARMA 模型之前必须要设置最小规则支持度的值，系统默认的是 20%，这个值的大小直接关系到最后生成的规则的数目。注意，在 Apriori 节点中设置的是“最低条件支持度”，是规则前项的支持度。而这里设置的是整个规则的支持度，是同时包含了规则前项和规则后项的支持度。这里设置为“15.0”。

最小规则置信度（Minimum Rule Confidence）：置信度定义为使用指定的规则对整个训练集中包含该规则前项部分的样本预测其后项的正确率。系统默认是 20%。如果得到的规则太多，尝试提高该项设置，如果得到的规则太少（或者根本就没有规则），尝试降低该项设置。这里将最低规则置信度设置为“50.0”。

最大规则大小（Maximum Rule Size）：设置每个规则中最多能包含多少个项。当只需要比较短的规则时，可以通过减小 Maximum Rule Size 来加速建模。

以上参数的设置对建模的结果影响很大，在实际操作中可以根据建模的结果重新调整这些参数，并重新建模，直到得到满意的结果。

切换到编辑窗口的“专家”标签下，可对建模进行高级设置，如图 5.17 所示。

将“模式”选项设置为“专家”，即可进行高级设置。

排除具有多个后项的规则（Exclude Rules With Multiple Consequents）：如果选中该选项，将排除那些规则后项含有多个项的规则。例如规则 bread∧cheese∧fish ⇒ wine∧fruit，由于规则的后项含有两个项，所以这条规则将被舍弃。



图 5.17　CARMA 模型高级设置

设置修剪值（Set Pruning Value）：为了节省内存，CARMA 算法会周期性剔除当前非频繁的项集。这就是在 CARMA 的 PhaseI 的第 3 步中，每处理完 k 个事务就检查 V 中的每一个项集并将非频繁者删除。选择这个选项可以设定 CARMA 执行这个算法的周期，如果该值设定的比较小，可能可以减少该算法所占用的内存，但是可能会增加建模的时间；反之，如果该值设定的比较大，那么可能会增大该算法所占用的内存，但是可以减少建模所需要的时间。系统设定的默认值为 500。

改变支持（Vary Support）：CARMA 允许在遍历事务数据库的过程中随时改变支持度阈值。然而，在算法运行过程中由用户自行给出各个支持度阈值是不现实的。在 Clementine 的 CARMA 算法实现中，允许支持度阈值取 4 个值，分别记为 s_1、s_2、s_3、s_4。初始值通常较大，然后逐渐减小。s_1 用于处理第 1～9 个事务，s_2 用于处理第 10~99 个事务，s_3 用于处理第 100～4999 个事务，s_4 用于处理第 4999 个以后的事务，它就是用户在“模型”标签下设置的“最低规则支持度”。根据事务数据库中的事务数量 t 的不同，4 个支持度阈值从大到小的下降速度不同：

- 如果最低规则支持度 $s\geq0.2$ 或者 $t<19$，则设置 $s_1=s_2=s_3=s_4$。
- 如果 $19\leq t<190$，则设置 $s_1=5s_2$，$s_3=s_4=s_2$，那么总的支持度阈值 $s=\dfrac{9s_1+(t-9)s_2}{t}$。
- 如果 $190\leq t<7000$，则设置 $s_1=5s_2$，$s_2=2s_3$，$s_4=s_3$，那么总的支持度阈值 $s=\dfrac{9s_1+90s_2+(t-99)s_3}{t}$。
- 如果 $t\geq7000$，则设置 $s_1=5s_2$，$s_2=2s_3$，$s_3=5s_4$，那么总的支持度阈值 $s=\dfrac{9s_1+90s_2+4900s_3+(t-4999)s_4}{t}$。

在以上几种情况中，如果根据设置的“最低规则支持度”计算后使得 $s_1>0.5$，那么就将 s_1 设置为 0.5，其他阈值也做相应的调整，从而使得 $\dfrac{\sum_{i=1}^{n}s(i)}{t}=s$ 成立，其中 $s(i)$是处

理第 i 个事务时的支持度阈值（是 s_1、s_2、s_3 或者 s_4 中的某一个）。

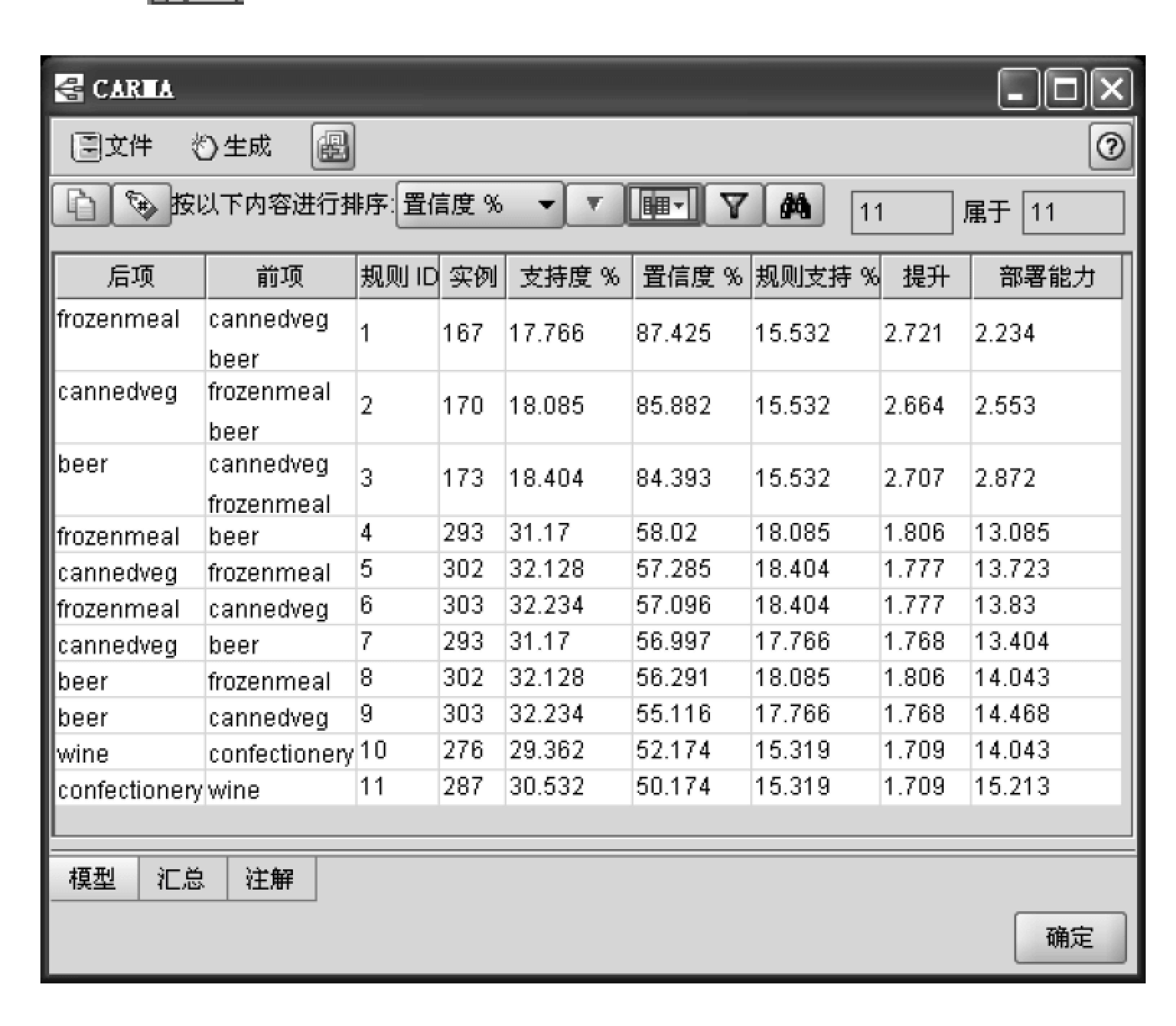
“改变支持”选项还有一个子选项“**估计的交易数**”（**Estimated Number of Transactions**），也就是事务数据库中的事务数量，通过设定该选项的值来，从以上 4 种情况中选择对应的计算方法，控制阈值减小的速度。这里输入事务数量 1000。

允许没有前项的规则（Allow Rules Without Antecedents）：选择该选项，可以允许规则集中出现没有前项的规则。在某些情况下，这个选项是很有用的，比如，如果想知道的就是顾客最经常购买的商品或商品组合，那么选择该选项就可以得到想要的结果。系统默认不使用该选项。

以上选项设置完毕后，单击“执行”按钮，即可在管理器窗口的“模型”标签下生成 CARMA 模型。

2. 浏览模型

对生成的模型进行浏览，可以看到所有的规则及其相关信息。右击“模型”标签下生成的 CARMA 模型，在快捷菜单中选择“浏览”命令，即可打开模型对话框，在对话框的工具栏中单击按钮并选择“显示全部”，如图 5.18 所示。



后项	前项	规则 ID	实例	支持度 %	置信度 %	规则支持 %	提升	部署能力
frozenmeal	cannedveg beer	1	167	17.766	87.425	15.532	2.721	2.234
cannedveg	frozenmeal beer	2	170	18.085	85.882	15.532	2.664	2.553
beer	cannedveg frozenmeal	3	173	18.404	84.393	15.532	2.707	2.872
frozenmeal	beer	4	293	31.17	58.02	18.085	1.806	13.085
cannedveg	frozenmeal	5	302	32.128	57.285	18.404	1.777	13.723
frozenmeal	cannedveg	6	303	32.234	57.096	18.404	1.777	13.83
cannedveg	beer	7	293	31.17	56.997	17.766	1.768	13.404
beer	frozenmeal	8	302	32.128	56.291	18.085	1.806	14.043
beer	cannedveg	9	303	32.234	55.116	17.766	1.768	14.468
wine	confectionery	10	276	29.362	52.174	15.319	1.709	14.043
confectionery	wine	11	287	30.532	50.174	15.319	1.709	15.213

图 5.18　生成的 CARMA 模型

可以看到，共生成了 11 条规则。规则的各参数的含义与第 5.2.2 节中所阐述的完全相同，这里不再赘述。

另外，在模型对话框的“汇总”标签下，可以看到关于本次建模的信息概要。其中“有效事务数”为 940（全部事务数为 1000）。通过对原始数据的分析，不难发现，在 1000 个事务中，有 60 个事务所有的输入变量值均为“F”，也就是说有 60 个顾客什么都没有

买，所以 CARMA 算法将它们过滤掉了。注意，Apriori 算法没有这个功能。

3. 生成节点

在生成的 CARMA 模型中可以通过“生成”（Gengerate）菜单生成不同的节点：选择节点（Select Node）、规则集（Rule Set）以及已过滤的模型（Filtered Model），这些节点可以被加入到数据流中以对数据进行相应功能的处理。

（1）选择节点

选中某条想要研究的规则，单击“生成”菜单的“选择节点”命令生成一个选择节点，可以将其加入到数据流中用于筛选需要的记录，其筛选的条件就是根据规则的前项来构造的。比如选中图 5.18 中的第一条记录，生成选择节点，则该节点可以筛选出所有购买了 cannedveg 和 beer 的记录，如图 5.19 所示。

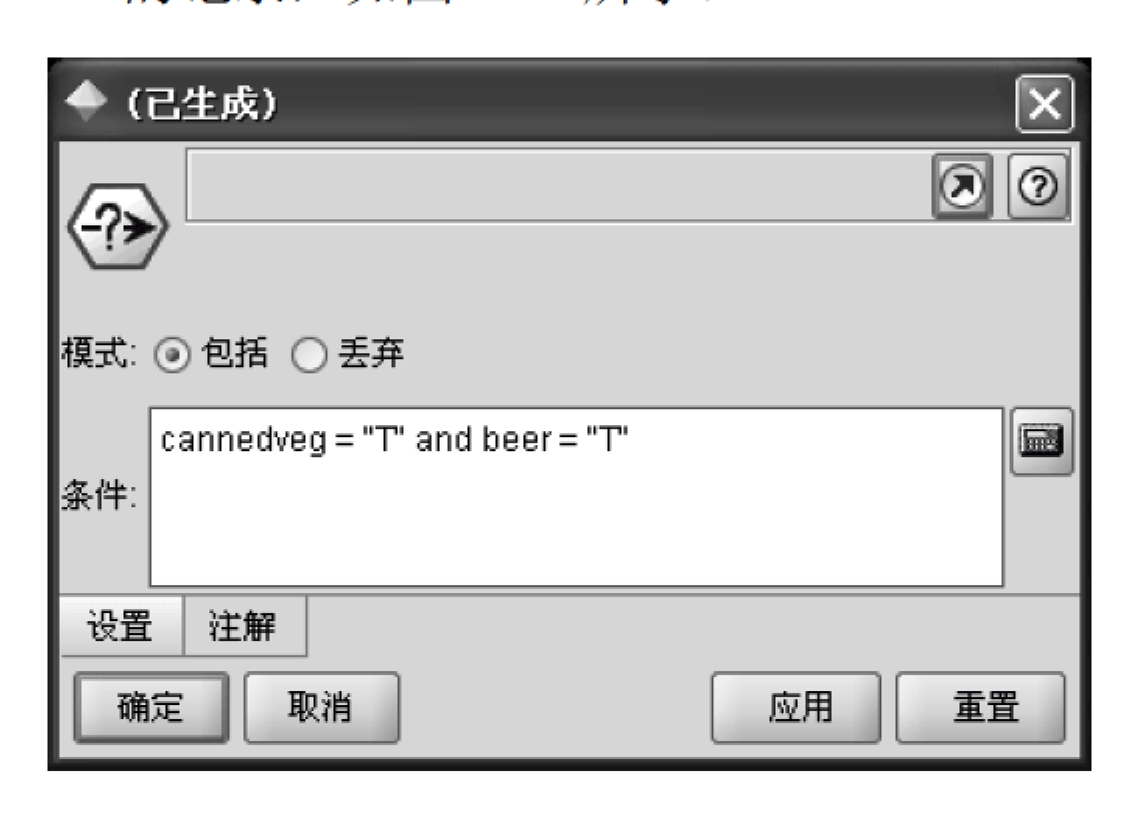


图 5.19　生成的选择节点

在图 5.19 所示窗口的“注解”标签下，可以对该节点进行命名。

（2）规则集

单击“生成”菜单的“规则集”命令，可以指定一个项（目标字段），从规则集中将那些规则后项包含了目标字段的规则全部提取出来，生成一个规则集节点。进而将这个规则节点放入数据流中对每行记录是否含有目标字段进行预测。

例如想研究哪些客户有可能会购买 cannedveg。单击“生成”菜单的“选择节点”命令，打开“生成规则集”窗口，做如图 5.20 所示的设置。



图 5.20　生成规则集节点

规则集名称（Rule Set Name）：为生成的规则集指定一个名称，如“cannedveg”。

在以下位置创建节点（Creat Node On）：指定生成的规则节点放置的位置。“工作区”（Canvas）就是放在数据流区域中，“GM 选项板”就是放在右上角的模型窗口内。

目标字段（Target Field）：这个选项是十分重要的，它指明需要预测的字段，进而根据这个字段来从 CARMA 的规则集中提取出相应的对这个字段进行预测的规则。

最低支持（Minimum Support）：这个选项为选择规则设定一个阈值，只选择那些支持度大于设定值的规则。注意，这里的支持度是指规则支持度。

最低置信度（Minimum Confidence）：对规则的置信度设定一个阈值，只选择那些置信度大于设定值的规则。

默认值（Default Value）：设定一个预测的默认值。即规则节点判断某个样本不是指定的目标字段值时就预测为 Default value。这里设定为 F。

以上设置完成后，单击“确定”按钮，即可在数据流区域生成 cannedveg 节点。双击该节点，打开浏览窗口，单击工具栏的按钮，并选择“完全展开所有分支”可以浏览该节点的详细设置，如图 5.21 所示。



图 5.21　浏览规则集节点

可以看到，该规则集包含了图 5.18 中的 3 个规则，它们的 ID 号分别为 2、5、7。

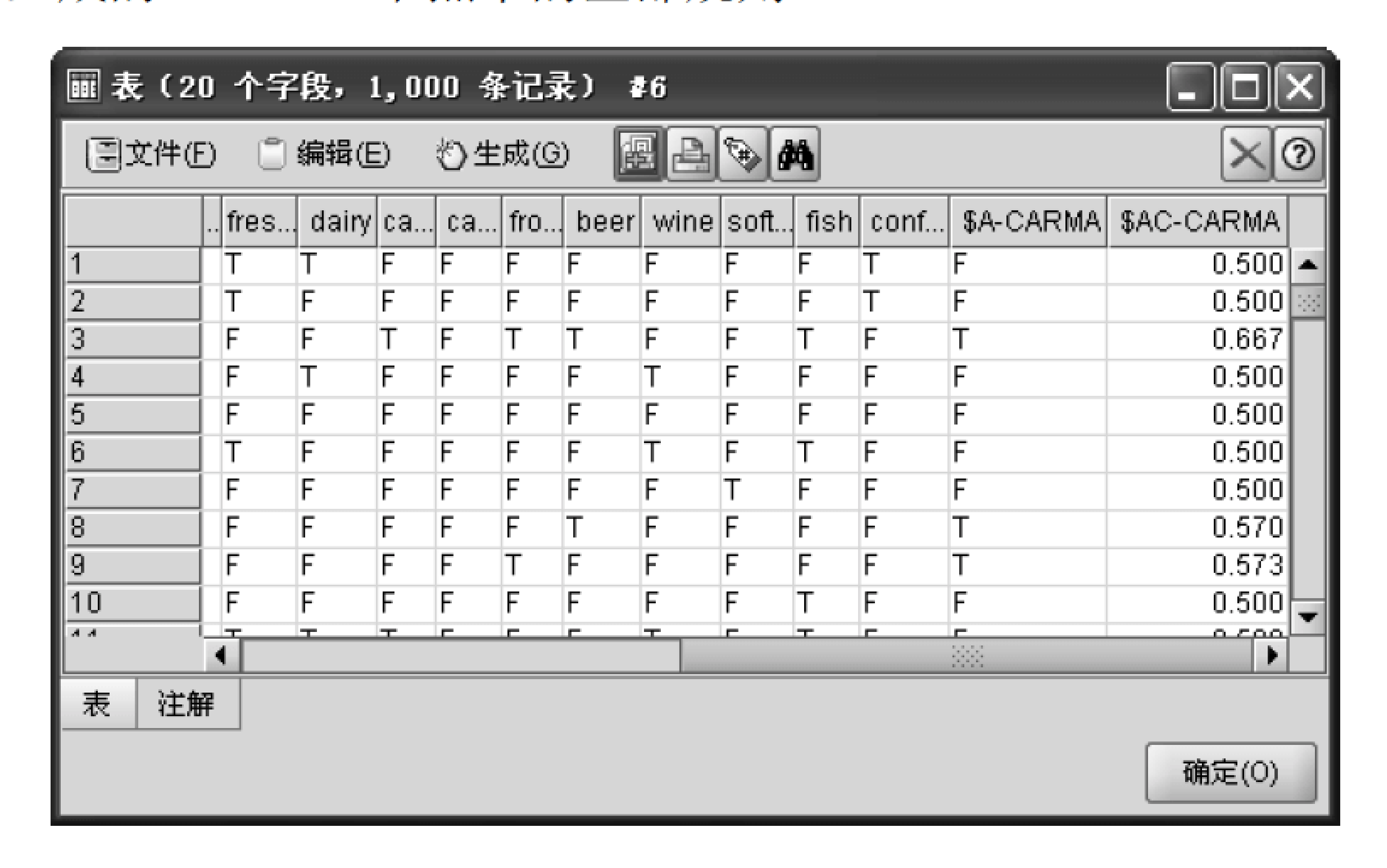
建立从“类型”节点到 cannedveg 节点的连接，并在 cannedveg 节点后添加“表”节点，执行该“表”节点，可以得到 cannedveg 节点对数据集中每个样本的预测结果，如图 5.22 所示。

表中最后两列即为该规则节点的预测结果，其中$A-CARMA 为预测的结果，$AC-CARMA 为该预测的置信度。

（3）已过滤的模型

单击“生成”菜单的“已过滤的模型”命令，可以生成一个 CARMA 模型，该模

型包含了生成的 CARMA 节点中的全部规则。



	..	fres...	dairy	ca...	ca...	fro...	beer	wine	soft...	fish	conf...	$A-CARMA	$AC-CARMA
1		T	T	F	F	F	F	F	F	F	T	F	0.500
2		T	F	F	F	F	F	F	F	F	T	F	0.500
3		F	F	T	F	T	T	F	F	T	F	T	0.667
4		F	T	F	F	F	F	T	F	F	F	F	0.500
5		F	F	F	F	F	F	F	F	F	F	F	0.500
6		T	F	F	F	F	F	T	F	T	F	F	0.500
7		F	F	F	F	F	F	F	T	F	F	F	0.500
8		F	F	F	F	F	T	F	F	F	F	T	0.570
9		F	F	F	F	T	F	F	F	F	F	T	0.573
10		F	F	F	F	F	F	F	F	T	F	F	0.500

图 5.22　规则集的预测结果

4. 用模型对数据进行预测

CARMA 模型可以直接放在数据流中对数据进行打分预测（scoring）。将生成的 CARMA 模型拖入数据流区域，建立从“类型”节点到该节点的连接，双击该节点，在“设置”标签下，对模型进行相关的参数设置，如图 5.23 所示。



图 5.23　设置关联规则模型

最大预测数（Maximum Number of Predictions）：该选项被用来设定对每一行数据所作预测的最大个数，因为对于每一个数据行都可以使用很多规则对其进行预测得出很多不同的结果，该选项可以对所作预测的数据做限定，使系统仅输出置信度、支持度或者规则支持度等最大的前几个结果，该项一般和下面的“规则标准”结合使用，这里设

置为 2。

规则标准（Rule Criterion）：该选项选择将预测结果作排序的标准，标准包括：置信度、支持度、规则支持度、提升以及 D 部署能力，这里选择“置信度”。

允许重复预测（Allow Repeat Predictions）：设定是否允许输出相同的预测值，因为通常都有很多条规则的后项是一样的，选择该选项就可以允许系统使用多条规则进行预测，并且输出相同的结果。

忽略不匹配篮项目（Ignore Unmatched Basket Items）：选择该项可以在每一行包含的项目比规则中的前项要多时，仍然可以使用该规则进行预测。

检查预测不在篮中（Check That Predictions Are Not In Basket）：选定该选项，当一个规则的后项被包含在数据行中时，则不用该规则来预测该数据行。比如，用规则对数据行进行打分的目的是为了给顾客推荐想买的家具，那么如果该名顾客已经买了一个餐桌的话，那么他基本上不可能会再买餐桌了，此时如果再给他购买餐桌的建议就显得没有意义了。选定该项可以避免这种情况的发生。

检查预测在篮中（Check That Predictions Are In Basket）：选择此选项可确保规则后项也存在于数据行中。

不检测篮子中是否存在预测值（Do Not Check Basket For Predictions）：选择该选项可以在打分时使用所有的规则。

以上设置完毕后，在生成的 CARMA 模型节点后添加“表”节点，建立连接并执行之，所得结果如图 5.24 所示。

表（24 个字段，1,000 条记录） #1

文件(F) 编辑(E) 生成(G)

	tionery	$A-CARMA-1	$AC-CARMA-1	$A-Rule_ID-1	$A-CARMA-2	$AC-CARMA-2	$A-Rule_ID-2
1		wine	0.522	10	$null$	$null$	$null$
2		wine	0.522	10	$null$	$null$	$null$
3		$null$	$null$	$null$	$null$	$null$	$null$
4		confectionery	0.502	11	$null$	$null$	$null$
5		$null$	$null$	$null$	$null$	$null$	$null$
6		confectionery	0.502	11	$null$	$null$	$null$
7		$null$	$null$	$null$	$null$	$null$	$null$
8		frozenmeal	0.580	4	cannedveg	0.570	7
9		cannedveg	0.573	5	beer	0.563	8
10		$null$	$null$	$null$	$null$	$null$	$null$
11		frozenmeal	0.571	6	beer	0.551	9
12		$null$	$null$	$null$	$null$	$null$	$null$
13		cannedveg	0.573	5	beer	0.563	8
14		$null$	$null$	$null$	$null$	$null$	$null$
15		$null$	$null$	$null$	$null$	$null$	$null$
16		cannedveg	0.859	2	$null$	$null$	$null$

表 注解

确定(O)

图 5.24 模型的预测结果

可以看到，在表的最后显示了 6 列数据，这就是预测结果的置信度最高的两条规则所预测的结果。其中$A-CARMA 是预测值，$AC-CARMA 是置信度，$A-Rule_ID 是预

测时所使用的规则的 ID 号。

5.4 序列模式

就购物篮数据而言，前面的 Apriori 算法和 CAMAR 算法用于发现客户在购物时商品之间的关联关系。序列模式挖掘要发现的是事件在发生过程中的先后顺序上的规律，例如客户在多次购物活动中购买商品的顺序模式。Rakesh Agrawal 在提出序列的概念时，举了一个形象的例子。比如一个顾客在租借影碟时，先租借“星球大战”，然后是“帝国反击战”，最后是“杰达武士归来”（3 部影片是以故事发生的时间先后而情节连续的）。不管顾客在任意两部影片之间是否还租借了其他影片，就这 3 部影片来说，顾客在租借时是有先后顺序的，即在租借了前两部影片之后，他租借第 3 部影片的概率是比较高的。这就是一个顾客在租借影片时的序列模式。序列模式挖掘正是要找到这样的规律。

5.4.1 序列与序列模式

给定一个客户交易事务数据库 *D*，其中的每一个事务由这样的属性字段组成：顾客 ID、交易时间，以及每次交易时间所购买的商品，如表 5-7 所示。

表 5-7 客户交易事务数据库 *D*

顾客 ID	Time1	Time2	Time3	Time4
1	cheese , crackers	wine	beer	-
2	wine	beer	cheese	-
3	bread	wine	cheese , beer	-
4	crackers	wine	beer	cheese
5	beer	cheese , crackers	bread	-
6	crackers	bread	-	-

一个顾客在某个时间点最多只有一次购物行为，或者没有购物行为（在表 5-7 中用“-”来表示）。比如某个顾客在购买了啤酒和面包之后，刚刚走出超市，又想起来没有买牛奶，于是他又返回超市购买了牛奶，那么这两次购物行为应该作为一次来对待。

每一种商品就是一个项目（item，简称“项”），因此一个顾客的一次购物行为就产生了一个项集，它是一个非空的集合。例如顾客 1 在 Time1、Time2 和 Time3 分别进行了一次购物行为，产生了 3 个项集{ cheese , crackers }、{wine}、{beer}。

序列，就是一个或多个项集有序地排列后组成的列表。例如，顾客 6 产生了这样一个序列：[{crackers} > {bread}]。符号“>”代表了一种顺序，表示其左边的项集产生之后才产生了其右边的项集。用$[s_1>s_2>\cdots>s_n]$来表示序列 *S*，其中 s_j 代表一个项集。序列中包含的项集的数量称为该序列的长度（length）。例如[{crackers} > {bread}]的长度为 2。长度为 *k* 的序列称为“*k*-序列”。

对于序列 $A=[a_1>a_2>\cdots>a_n]$和序列 $B=[b_1>b_2>\cdots>b_m]$，如果存在一组整数 $i_1<i_2<\cdots<i_n$，使得 $a_1\subseteq b_{i_1}$，$a_2\subseteq b_{i_2}$，…，$a_n\subseteq b_{i_n}$ 成立，则称序列 *A* 包含于序列 *B*。例如，序列

[{cheese }>{wine}>{beer}]包含于[{bread, cheese}>{wine, crackers }>{cheese}>{beer}]，这是因为，{cheese } ⊆ {bread, cheese}，{wine} ⊆ {wine, crackers}，{beer} ⊆ {beer}。

在一个序列集中，如果某个序列 *s* 不包含于任何其他序列中，则称 *s* 是“极大序列”(Maximal Sequence)。

将一个顾客每次购物产生的项集按照时间顺序排列，就形成了该顾客的事务序列。对于一个序列 *s*，如果 *s* 包含于顾客 *C* 的事务序列，就称“顾客 *C* 支持序列 *s*”。那么，支持序列 *s* 的顾客数量除以顾客总数量，就是序列 *s* 的支持度。

例如序列[{crackers} > {wine} > {beer}]，在 6 个顾客中，顾客 1 和顾客 4 支持该序列，所以其支持度为 2/6=33.3%。

在进行数据挖掘时，由用户指定一个最小支持度阈值。把那些支持度大于等于这个阈值的序列称为“频繁序列”。长度为 *k* 的频繁序列记做“频繁 *k*-序列”。

给定一个事务数据库 *D*，那么序列模式挖掘就是要从数据中找出所有的频繁序列，并从中取出那些极大序列，每一个这样的序列都代表了一个**序列模式**。

一个序列模式由两个部分组成，即前件序列（Antecedent Sequence）和后件序列（Consequent Sequence）。对于一个极大频繁序列 $S=[s_1>s_2>\cdots>s_n]$，序列$[s_1>s_2>\cdots>s_{n-1}]$构成了序列模式的前件，$[s_n]$就是序列模式的后件，可以表示为：

If $[s_1>s_2>\cdots>s_{n-1}]$ then $[s_n]$，或者 $[s_1>s_2>\cdots>s_{n-1} => s_n]$

例如，给定最小支持度阈值为 20%，序列[{crackers} > {wine} > {beer}]是从表 5-7 中挖掘出来的极大频繁序列，因为该序列的支持度大于 20%且不包含于其他频繁序列。然而序列[{crackers} >{wine}]虽然也是频繁的，但却不是极大的。对于序列[{crackers} > {wine} > {beer}]，可以生成一个序列模式[{crackers} > {wine} => {beer}]，表示“顾客如果先购买了 crackers，然后又购买了 wine，那么他将很可能在未来的某个时刻购买 beer”。

一个序列模式的置信度，是指同时支持序列前件和后件的顾客数量除以支持序列前件的顾客数量所得到的比值。

5.4.2 序列模式挖掘算法

1. 序列模式挖掘算法介绍

目前的序列模式挖掘算法大多都是 Apriori 类算法的改进，如 AprioriAll、AprioriSome 和 GSP 算法等。

AprioriAll 算法与 Apriori 类似，首先遍历数据库生产候选序列并利用 Apriori 的特性进行剪枝来得到频繁序列。每次遍历时通过连接上一次得到的频繁序列来生成新的长度加 1 的候选序列。然后对每个候选序列进行扫描，按照最小支持度来确定哪些序列是频繁序列。AprioriAll 算法的不足在于容易生成数量庞大的候选序列，同时还需要多次扫描数据库。

AprioriSome 与 AprioriAll 只是在序列阶段有所不同，AprioriAll 是首先生成所有的频繁序列，然后在极大序列阶段删除那些非极大的序列。AprioriSome 将序列分成两个部分分别计数，前半部分只对一定长度的序列计数，后半部分跳过已经计数的序列。在实

际过程中两个部分是混合在一起的，以减少候选序列占用的资源。

GSP 算法是 AprioriAll 的扩展算法，其算法的执行过程和 AprioriAll 类似，最大的不同就在于 GSP 引入了时间约束、滑动窗口和分类层次技术，增加了扫描的约束条件，有效地减少了需要扫描的候选序列的数量，同时还克服了基本序列模型的局限性，更切合实际，减少多余的无用模式的产生。另外 GSP 利用哈希树来存储候选序列，减小了需要扫描的序列数量。

2. AprioriAll 算法

在 Rakesh Agrawal 关于序列模式挖掘的论述中，将序列模式挖掘的一般步骤分为 5 个阶段，即排序阶段、频繁项集阶段、转换阶段、序列阶段和选极大序列阶段。

（1）排序阶段

在排序阶段对数据库中的记录进行排序，以顾客 ID 为主键、购物时间为次键进行升序排列。这样排序之后，对于单个的顾客来说，实际上得到了每个顾客的购物序列，称为顾客序列（Customer Sequnence）。

例如对表 5-8 所示的顾客购物数据库 *D* 进行排序之后得到表 5-9。

表 5-8　原始数据库 *D*

购物时间 Transaction Time	顾客 ID Customer Id	所购商品代码 Items Bought
June 10 '93	2	10, 20
June 12 '93	5	90
June 15 '93	2	30
June 20 '93	2	40, 60, 70
June 25 '93	4	30
June 25 '93	3	30, 50, 70
June 25 '93	1	30
June 30 '93	1	90
June 30 '93	4	40, 70
July 25 '93	4	90

表 5-9　排序之后的数据库

购物时间 Transaction Time	顾客 ID Customer Id	所购商品代码 Items Bought
June 25 '93	1	30
June 30 '93	1	90
June 10 '93	2	10, 20
June 15 '93	2	30
June 20 '93	2	40, 60, 70
June 25 '93	3	30, 50, 70
June 25 '93	4	30
June 30 '93	4	40, 70
July 25 '93	4	90
June 12 '93	5	90

进而可得到顾客序列数据库（如表 5-10 所示）。

表 5-10　顾客序列数据库

Customer Id	Customer Sequence
1	[{30},{90}]
2	[{10,20},{30},{40,60,70}]
3	[{30, 50, 70}]
4	[{30},{40,70},{90}]
5	[{90}]

（2）频繁项集阶段

设定支持度阈值后，在频繁项集阶段要找到所有频繁项集组成的集合 L。同时也得到了所有的频繁 1-序列，因为每个频繁 1-序列实际上就是由一个频繁项集构成的。

为了便于数据处理，通常将所有的频繁项集映射到一些连续的整数上。例如这里找到的频繁项集为{30}、{40}、{70}、{40,70}、{90}，那么可以如表 5-11 所示来映射这些频繁项集。

表 5-11　频繁项集映射为整数

频繁项集	映射整数
{30}	1
{40}	2
{70}	3
{40,70}	4
{90}	5

（3）转换阶段

在这一阶段，要在前两个阶段的基础上对顾客序列做进一步的转换。由于在寻找频繁序列的过程中要不断检测一个给定的序列是否被包含在一个顾客序列中（即检测一个顾客是否支持给定的序列），本阶段的转换将提高这一检测的速度。

转换的过程是这样的：对于顾客序列数据库中的每个序列 t（即每个记录），用该序列所包含的所有频繁项集来代替。如果序列 t 中有某个项集 s，s 的全部子集都不是频繁项集，那么 s 将被从序列 t 中剔除；如果序列 t 中不包含任何频繁项集，那么在转换之后它将不被保留。但在计算顾客总数时，它仍将被计算在内。转换之后得到的数据库记做 D_T。

例如表 5-10 在被转换后得到的 D_T 如表 5-12 所示。

表 5-12　转换后的序列数据库 D_T

Id	Original Customer Sequence	Transformed Customer Sequence	After Mapping
1	[{30},{90}]	[{30},{90}]	[1,5]
2	[{10,20},{30},{40,60,70}]	[{30},{{40},{70},{40,70}}]	[1,(2,3,4)]
3	[{30, 50, 70}]	[{30, 70}]	[(1, 3)]
4	[{30},{40,70},{90}]	[{30},{{40},{70},{40,70}},{90}]	[1,(2,3,4),5]
5	[{90}]	[{90}]	[5]

例如 Id 号为 2 的顾客序列在进行转换时，项集{10,20}由于子集都不是频繁项集而被从序列中剔除。项集{40,60,70}则被其所包含频繁项集{{40},{70},{40,70}}所取代。

（4）序列阶段

在转换后的序列数据库的基础上，本阶段利用频繁序列挖掘算法寻找频繁序列。

AprioriAll 算法描述如下：

输入：频繁 1-序列，顾客序列数据库 D_T

输出：极大序列集

算法：

```
    L_1={频繁 1-序列}；
    for (k=2; L_{k−1} ≠ ∅ ; k++) do
        {C_k=aprioriall-generate(L_{k−1})；    //根据频繁(k−1)-序列产生候选 k 序列
        for (D_T 中的每个序列 c)   do
            对 C_k 中凡是被 c 包含的候选者，其计数加 1；  //计算支持计数
            L_k=C_k 中所有满足最小支持度的候选者；}
     Answer=在 ⋃_k L_k 中的极大序列；
```

函数 aprioriall-generate(L_{k-1})的定义：

参数：频繁(k−1)-序列集 L_{k-1}

返回值：候选 k-序列集 C_k

功能：对 L_{k-1} 进行自连接，然后剪枝

```
function aprioriall-generate(L_{k−1})
    {insert into C_k
    select   p.litemset_1, p.litemset_2,⋯, p.litemset_{k−1}, q.litemset_{k−1}
    from     L_{k−1} as p,   L_{k−1} as q          //litemset 表示频繁项集
    where    p.litemset_1 = q.litemset_1,⋯, p.litemset_{k−2} = q.litemset_{k−2}; //连接
    for (C_k 中的每个序列 c ) do
        {for (c 的每个(k−1)子序列 s) do
            if s ∉ L_{k−1} then
                delete s from C_k ;}     //剪枝，其原理是：如果一个序列的任一子序
                                         //列不频繁，则这个序列也不频繁
```

（5）选极大序列阶段

根据上一阶段得到的结果，从频繁序列中筛选出极大序列。

令频繁序列集为 S，S 中的序列长度的最大值为 n，那么下面这个算法可以用来筛选出全部极大序列：

```
for ( k = n; k > 1; k− −)   do
    {for S 中的每个 k-序列 s_k  do
        从 S 中删除 s_k 的所有子序列；}
```

下面利用上述原理来对表 5-12 中映射后的顾客序列进行频繁序列挖掘。设支持度阈值为 40%，即 5 个顾客中至少有两个顾客支持的序列为频繁序列（支持计数为 2）。首先根据频繁 1-项集生成频繁 1-序列，然后通过若干次连接和剪枝，得到所有频繁序列，整

个过程如图 5.25 所示。

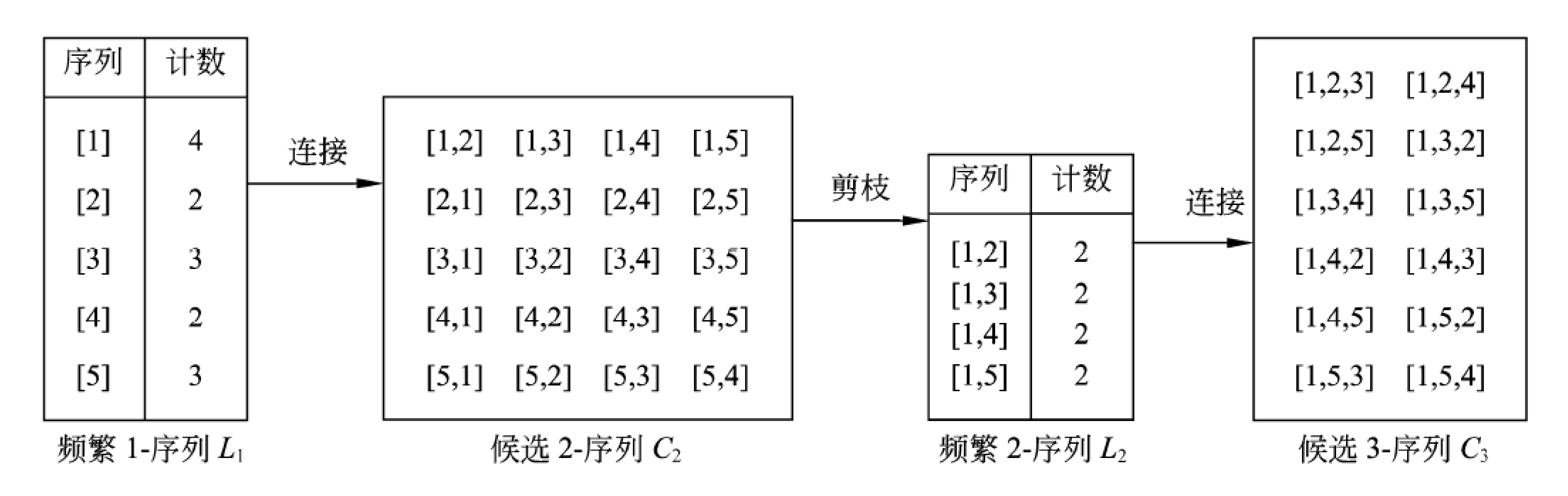


图 5.25　频繁序列挖掘示例

其中，候选 3-序列 C_3 中的所有序列的支持计数均为 0，算法停止。

注意，在剪枝步骤中包括两个内容。其一是将候选集中支持度小于阈值的序列删除，在计算一个序列的支持度时，一定要注意“包含”的定义。当一个候选序列被事务序列所包含时，其支持计数才加 1。例如序列[1,3]被[1, (2,3,4)]包含，但不被[(1,3)]包含。其二是将子序列不频繁的序列删除。

到此为止，所有的频繁序列已经找到，即 $L = L_1 \cup L_2$ 。然后从中找出极大序列。将长度较大的序列的所有子序列从 L 中删除，这里将[1]、[2]、[3]、[4]、[5]删除，最后得到的极大频繁序列有[1,2]、[1,3]、[1,4]、[1,5]。根据表 5-11，将整数映射回所代表的商品代码，最终得到的序列模式为：

[{30}>{40}]；　[{30}>{70}]；　[{30}>{40, 70}]；　[{30}>{90}]

5.4.3　在 Clementine 中应用序列模式挖掘

在 Clementine 中有一个“序列”节点（Sequence Node）用于序列模式的建模，它基于 CARMA 关联规则算法，该算法使用一个有效的两次传递方法查找序列。

在利用“序列”节点创建序列模式集时，需要指定一个 ID 字段和一个可选的时间字段，以及一个或多个内容字段。这些设置必须在建模节点的“字段”选项卡上进行；不能从上游类型节点中读取。该 ID 字段可以是任意方向或任意类型。如果指定时间字段，则该字段可以是任意方向，但必须是数字、日期、时间或时间戳。如果不指定时间字段，序列节点则会使用隐含的时间戳，实际上是使用行号作为时间值。内容字段可以是任意类型和任意方向，但是所有内容字段必须是相同的类型。如果这些字段是数字型的，则必须为整数范围（不是实数范围）。

这里，对某超市的顾客购物事务数据库进行分析以提取序列模式。从事务数据库中随机抽取 10 个顾客，每个顾客都有多次购物记录，组成训练数据集，共 67 个训练样本，存放在 sequence.xls 文件中。样本属性包括顾客 ID、购物时间以及商品名称，其中购物时间已经做了数值化处理，如表 5-13 所示。其中，“T”表示购买了某商品，“F”表示未购买某商品。

表 5-13　事务数据库

ID	Time	pasta	milk	water	biscuits	coffee	brioches	…	yoghurt
1	1	T	F	F	F	F	F	…	F
1	2	F	T	T	F	F	F	…	F
1	3	F	F	F	F	F	F	…	T
1	6	F	F	F	F	F	T	…	F
1	7	F	F	F	F	F	F	…	F
1	8	F	F	F	F	T	F	…	F
1	9	F	F	F	T	F	F	…	F
1	10	F	F	F	F	F	F	…	F
2	1	F	T	T	T	F	F	…	F
…	…	…	…	…	…	…	…	…	…
10	6	F	F	F	F	F	T	…	F

1. 构建模型

数据流如图 5.26 所示。

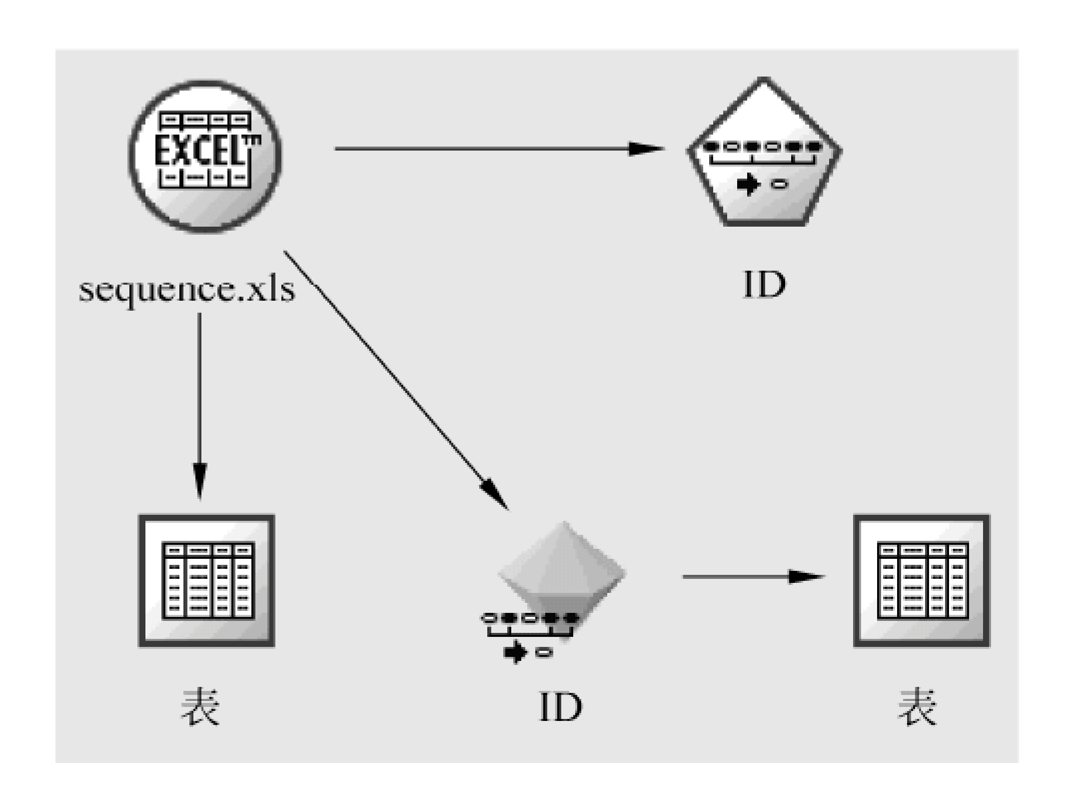


图 5.26　序列建模数据流

首先在数据流区域添加 Excel 数据源节点，并将 sequence.xls 加载到数据源节点。然后添加“表”节点，可以浏览数据集中的数据。

将“序列”节点添加到数据流中，建立由数据源节点到序列节点的连接，然后对“序列”节点进行设置，如图 5.27 所示。

在“序列”节点的编辑窗口中的“字段”标签下，首先对“ID 字段”进行设置。ID 字段的每个唯一值都应该表明一个特定的分析单元。例如，在市场购物篮的应用中，每个 ID 可能表示一个客户。对于 Web 日志分析应用，每个 ID 可能代表一个计算机（以 IP 地址表示）或一个用户（以登录数据表示）。这里，从右侧的下拉框中选择 ID 即可。

如果在数据中使用字段来表明事件时间，要选择“使用时间字段”并指定要使用的字段。时间字段必须是数字、日期、时间或时间戳型的。如果不指定时间字段，则假设记录按照从数据源出发的顺序到达，记录号将用做时间值（第一个记录发生在时间“1”；第二个记录发生在时间“2”；以此类推）。这里，选中“使用时间字段”复选框，然后从

右侧的下拉框中选择 Time 字段。



图 5.27　设置序列节点的字段属性

内容字段是参与建模的字段，这些字段包含与序列建模有关的事件。单击右侧的按钮，弹出“选择字段”对话框，从中选择内容字段。在选择时，按住 Shift 键不放可以同时选择多个连续的字段，按住 Ctrl 键不放可以同时选择多个不连续的字段。这里，选中全部字段。

切换到“模型”标签下，如图 5.28 所示。



图 5.28　建模基本设置

在这里可以对节点进行一些基本的设置，包括：

最小规则支持度（Minimum Rule Support）（%）：指的是训练数据中包含整个序列的ID的比例。这里设置为30%。

最小规则置信度（Minimum Rule Confidence）（%）：指的是得到正确预测的ID在所有使用规则进行预测的ID中所占的百分比。基于训练数据，该百分比的计算如下：包含整个序列的ID数量除以其中包含条件的ID数量。置信度低于指定标准的序列将被放弃。这里设置为50%。

最大序列大小（Maximum Sequence Size）：设置序列中不同项集的最大数量。如果相关序列相对较短，则可以降低此设置，以加快序列集构建速度。这里设置为10。

要添加到流的预测（Predictions To Add To Stream）：指定生成的结果模型节点要添加到流中的预测数量。这是指在将来用生成的序列模式节点来预测某个测试样本时显示多少个置信度最高的预测值。这里设置为3。

切换到“专家”标签下，如图5.29所示。



图5.29 建模高级设置

将“模式”选项设置为“专家”，可以进行如下高级设置：

设置最长持续时间（Set Maximum Duration）：如果选择了此选项，序列将被限制为小于或等于指定值的一个持续时间（第一个项集和最后一个项集之间的时间）。例如对于购物篮数据，对每个顾客而言，只选取其在所限制时间段里的事务。如果没有指定时间字段，该持续时间则以原始数据中的行数（记录数）表示。如果使用的时间字段为时间、日期或时间戳型字段，该持续时间则表示为秒数。对于数值字段，持续时间则使用与字段相同的单位数表示。

设置修剪值（Set Pruning Value）：为了节省内存，序列节点中使用的CARMA算法会在处理期间定期从其潜在项目集合列表中删除（修剪）不常使用的项集。选择此选项可调整修剪的频率。指定的数字决定了修剪频率。输入较小的值可降低该算法的内存要求（但可能会延长所需的训练时间），输入较大的值会加快训练速度（但可能会提高内

存要求)。

设置内存中的最大序列数(Set Maximum Sequences In Memory):如果选择了此选项,CARMA 算法则会将建模期间备选序列的内存限制为指定的序列数。如果 Clementine 在序列建模期间使用的内存过多,应选择此选项。此数字应该比最终模型中预期的序列数大很多。

抑制项目集之间的间距(Constrain Gaps Between Item Sets):通过此选项可以针对不同项集的时间间距指定约束。如果选择了此选项,则不会考虑时间间距小于所指定的“最小间距”或大于“最大间距”的项目集合作为序列的组成部分。使用此选项可避免考虑包括较长时间区间或者在很短的时间跨度内发生的那些序列。对于购物篮数据,这里的间距是指某个顾客的某两次购物在时间上的差。

以上设置完成后,单击“执行”按钮,即可生成序列模型节点,并显示在管理器窗口的“模型”标签下。

2. 浏览模型

右击生成的序列模型节点,在快捷菜单中选择“浏览”命令,打开浏览窗口,单击工具栏中的按钮,并选择“显示全部”,即可浏览到所有生成的序列模式,如图 5.30 所示。



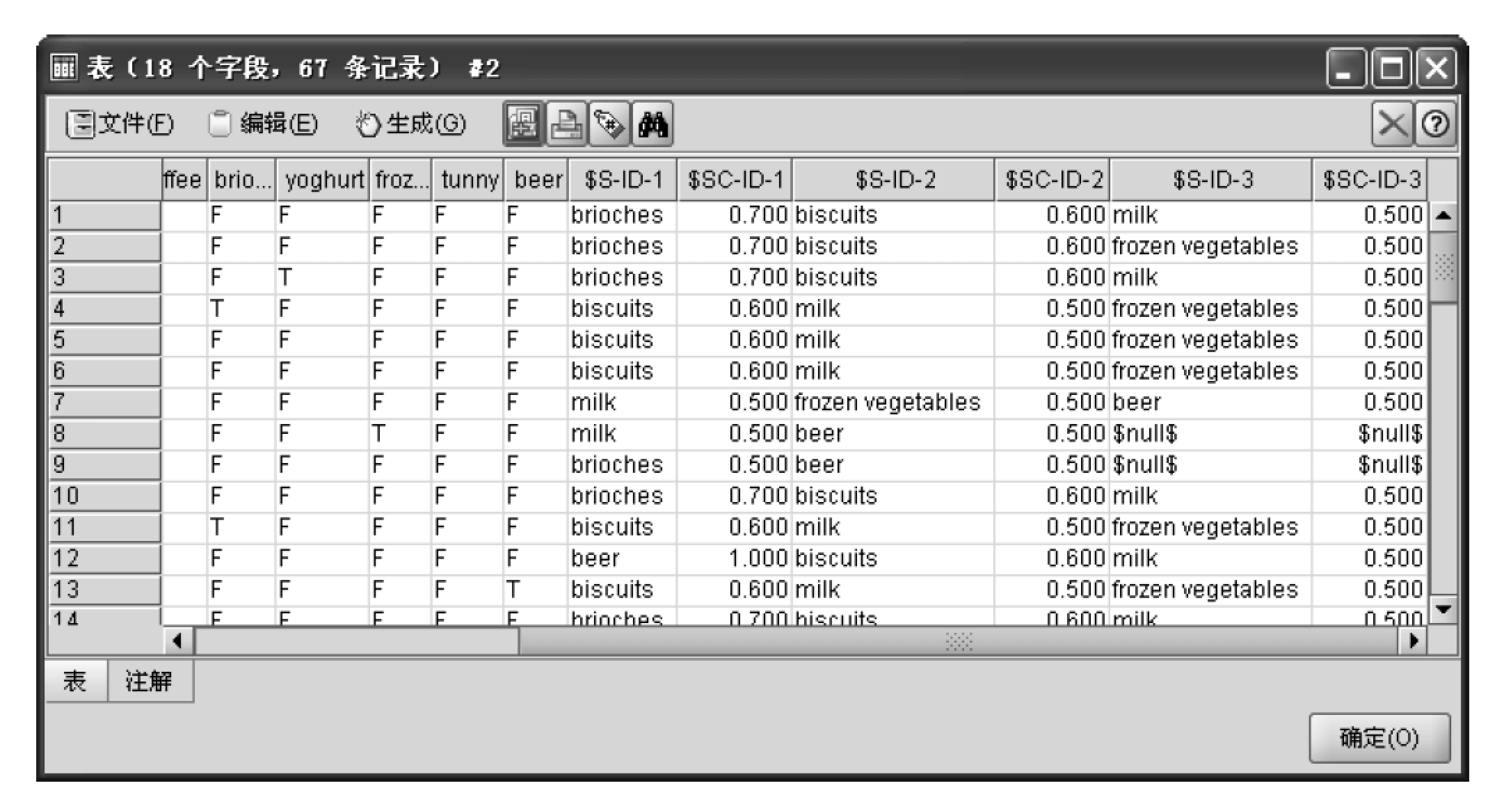
前项	后项	实例	支持度 %	置信度 %	规则支持 %
biscuits coffee	beer	3	30.0	100.0	30.0
pasta	brioches	7	100.0	70.0	70.0
pasta	biscuits	6	100.0	60.0	60.0
pasta	milk	5	100.0	50.0	50.0
pasta	frozen vegeta...	5	100.0	50.0	50.0
water	brioches	5	100.0	50.0	50.0
yoghurt	milk	3	60.0	50.0	30.0
coffee	beer	4	80.0	50.0	40.0
yoghurt	frozen vegeta...	3	60.0	50.0	30.0
milk	brioches	5	100.0	50.0	50.0
water	beer	5	100.0	50.0	50.0

图 5.30　生成的序列模式

3. 生成的序列模型用于预测

生成的序列模型节点可以添加到某个数据流中对数据进行预测。例如,直接将生成的模型添加到数据流中来对样本数据进行预测。

将生成的 ID 模型节点加入到数据流区域，并建立由 sequence.xls 节点到 ID 节点的连接，然后在 ID 节点后添加“表”节点并执行之，即可得到预测结果，如图 5.31 所示。



表（18 个字段，67 条记录） #2

文件(F) 编辑(E) 生成(G)

	ffee	brio...	yoghurt	froz...	tunny	beer	$S-ID-1	$SC-ID-1	$S-ID-2	$SC-ID-2	$S-ID-3	$SC-ID-3
1		F	F	F	F	F	brioches	0.700	biscuits	0.600	milk	0.500
2		F	F	F	F	F	brioches	0.700	biscuits	0.600	frozen vegetables	0.500
3		F	T	F	F	F	brioches	0.700	biscuits	0.600	milk	0.500
4		T	F	F	F	F	biscuits	0.600	milk	0.500	frozen vegetables	0.500
5		F	F	F	F	F	biscuits	0.600	milk	0.500	frozen vegetables	0.500
6		F	F	F	F	F	biscuits	0.600	milk	0.500	frozen vegetables	0.500
7		F	F	F	F	F	milk	0.500	frozen vegetables	0.500	beer	0.500
8		F	F	T	F	F	milk	0.500	beer	0.500	$null$	$null$
9		F	F	F	F	F	brioches	0.500	beer	0.500	$null$	$null$
10		F	F	F	F	F	brioches	0.700	biscuits	0.600	milk	0.500
11		T	F	F	F	F	biscuits	0.600	milk	0.500	frozen vegetables	0.500
12		F	F	F	F	F	beer	1.000	biscuits	0.600	milk	0.500
13		F	F	F	F	T	biscuits	0.600	milk	0.500	frozen vegetables	0.500
14		F	F	F	F	F	brioches	0.700	biscuits	0.600	milk	0.500

表 注解

确定(O)

图 5.31 预测结果

可以看到，在原有的数据基础上，结果中多了 6 列数据。这 6 列数据是用所有的序列模式来对某个样本进行预测时，置信度最高的前 3 个预测结果以及它们的置信度。之所以只显示 3 个预测结果，是因为在设置建模参数时，将“要添加到流的预测”设置为了 3，如果想减少或增加预测结果，可以改变该参数的设置。

例如图 5.31 中的第一个记录的预测结果：

$S-ID-1=brioches　　$SC-ID-1=0.700

$S-ID-2=biscuits　　$SC-ID-2=0.600

$S-ID-3=milk　　$SC-ID-3=0.500

其含义是：就该顾客的这次购物来看，其下次购物时购买 brioches 的概率是 70%，购买 biscuits 的概率是 60%，购买 milk 的概率是 50%。

对于可用预测数量小于所请求预测数量的记录，其没有的预测值和置信度显示为 $null$。

第 6 章　数 据 筛 选

数据筛选，主要用于在建立数据挖掘模型之前对原始数据库的处理。目的是从原始数据库中筛选出合适的属性和合适的样本来作为训练数据库，从而建立更加科学、准确的模型。

本章介绍两种数据筛选算法：特征选择和异常检测。两种算法用于不同的场合。在 Clementine 中由相应的建模节点来实现这两种算法。

在建立分类模型之前，通常可以通过特征选择来筛选出那些和目标属性相关度较高的属性来参与建模。

当研究异常情况时，可以通过异常检测先从原始数据库中筛选出那些异常样本，然后再通过其他方法对这些异常样本进行分析。

6.1 特征选择

6.1.1 特征选择算法概述

特征选择用于在建立分类模型或者预测模型之前，对原始数据库进行预处理。因为在原始数据库中，可能包括成百甚至上千个属性字段，从而需要花费大量的时间和精力来检查模型究竟应该包含哪些字段或变量，也就是确定哪些字段来参与建模。那些和我们要预测的属性并没有什么关系或者关系不大的属性就没有必要参与到建模过程中，比如作为主键的“样本号”字段。可以使用特征选择算法来选择那些最为重要的字段来参与建模。

在统计学中，可以从两个不同的侧面来考察两个变量之间的关联关系，即独立性和相关性。另外，根据变量类型的不同，具体采用的分析方法也不尽相同。

1. 基本概念

表 6-1 显示了 500 种原料的基本信息，包括“ID”、“地区来源”、“价格”、“生产日期”、“保质期”、“等级”等属性。

假设现在要根据这个数据库建立一个预测模型，利用这个模型，可以根据某种原料的“地区来源”、“价格”、“保质期”等属性来预测其“等级”的值（是“一级”、“二级”还是“三级”）。

我们把被预测的那个属性称为“**目标变量**”（**Target Variable**），这里“等级”就是目标变量。把用来预测“目标变量”值的属性称为“**预测变量**”（**Predictor Variable**），这里“地区来源”、“价格”、“保质期”等属性就是预测变量。

特征选择，就是要考察数据库中的“预测变量”与“目标变量”的关联程度，或者

说考察每个预测变量对最终预测结果的重要程度。这个重要程度用一个指数（importance）来表示。特征选择算法要计算每个预测变量的 importance 指数。

表 6-1 原料基本信息数据库

ID	地区来源	价格	生产日期	保质期	等级
1	甲地区	100	2009-11-8	2 年	一级
2	乙地区	120	2009-10-4	3 年	二级
3	丙地区	140	2009-4-3	5 年	三级
4	乙地区	125	2009-5-4	2 年	二级
5	丙地区	200	2009-5-4	3 年	一级
…	…	…	…	…	…
500	丙地区	140	2009-10-4	1 年	二级

2．算法步骤

特征选择算法包括以下 3 个步骤：

（1）**筛选（Screening）**：除去不重要或有问题的预测变量和记录，例如预测变量含有过多缺失值，或者预测变量的变化太大或太少而变得无用。

（2）**分级（Ranking）**：对所有预测变量根据其重要程度进行排序。

（3）**选择（Selecting）**：生成在后续模型中使用的功能子集，例如仅保留最重要的预测变量，过滤或排除所有其他预测变量。

后续小节将详细介绍这 3 个步骤。

6.1.2 筛选

在筛选阶段，要删除那些不适合参与建模的预测变量，以及一些有缺陷的样本。

（1）满足下列任一条件的预测变量将被删除：

① 变量的值全部缺失。

② 变量的全部取值都是固定不变的值（即全部相同）。

③ 代表样本号的字段。

（2）满足下列任一条件的样本将被删除：

① 目标字段的值缺失的样本。

② 所有预测字段的值都缺失的样本。

（3）根据用户的自行设置来删除预测变量：

① 有多于 m_1%的值缺失，则删除该变量。缺失值百分比很大的字段几乎不提供任何预测信息。

② 对于离散型的预测变量，如果有超过 m_2%的样本取同一个值，则删除该变量。例如，如果数据库中 95%的客户开同一类型的车，则此信息无助于区分客户。

③ 对于连续型的预测变量，如果其标准差小于 m_3%，则删除该变量。

④ 对于连续型的预测变量，如果其变异系数$|CV|<m_4$%，CV=标准差/均值。因为如果变异系数接近 0，则变量值的变异性就不高。

⑤ 对于离散型的预测变量，变量取值的个数大于总样本量的 m_5%。则删除该变量。这一标准用于筛选掉相对于记录总数而言具有过多类别取值的字段。如果很高百分比的类别只含有一个观测值，则该字段用处有限。例如，如果每名客户都戴不同的帽子，则此信息在建立行为模式模型时就不太可能有用。

这里 m_1、m_2、m_3、m_4、m_5 的大小由用户自行设定。

6.1.3 分级

在分级阶段，要考察每个预测变量对于预测目标变量值的重要程度，并用 importance 指数来表示这个重要程度。最后还要根据重要程度的大小对所有预测变量排序，也就是给每个预测变量指定“秩”(Rank)。“秩”是指根据某种大小次序排列后形成的顺序号。

变量之间相关性的分析方法，是以统计学中的假设检验为基本原理的，这里涉及了一个基本的概念——假设检验中的 p 值（p value）。

进行假设检验是以样本数据为依据的，如果样本的观察结果与假设的数值有很大的差异，我们就会怀疑原假设是否成立。

例如，我们提出原假设 H_0：新生儿体重的总体均值=3190g。那么如果原假设成立的话，根据正态分布，新生儿体重的总体均值大于等于 3210g 的概率应该只有 0.01242（我们把这个概率称为 p 值）。现在从总体中随机地抽取一部分样本出来计算样本均值，结果为 3215g，大于 3210g。这个过程表明，一个小概率（0.01242）事件，居然通过一次实验就发生了！然而统计学里的“小概率”原理告诉我们，发生概率很小的随机事件在一次实验中几乎是不可能发生的。那么现在一个小概率事件在一次实验中发生了，其原因只能是最初的原假设在很大程度上是不成立的。

因此，这里的 p 值，就是在原假设 H_0 为真的前提下，某个事件发生的概率（而在之后的一次随机实验中的确发生了）。显然，如果 p 值很小，就表示小概率事件在一次实验中发生了，而这违背了“小概率”原理，我们就有理由拒绝原假设。而且，p 值越小，拒绝原假设的理由就越充分。

所以，利用假设检验的原理进行变量的相关分析，我们可以首先提出原假设：

H_0：变量 X 与变量 Y 完全不相关

然后根据样本数据集来观测某个发生的事件，并计算该事件在 H_0 假设下发生的概率（即 p 值），如果 p 值很小，就有一定的理由拒绝 H_0 假设，或者说，变量 X 与变量 Y 在某种程度上是相关的。可见，变量 X 与变量 Y 的相关程度是可以用 p 值来度量的。p 值越小，变量 X 与变量 Y 越相关；反之，它们则越独立。

因此，预测变量相对于目标变量的重要程度（importance 指数）可以这样来计算：

$$\text{importance}=1-p$$

根据目标变量是离散型还是连续型，计算 importance 指数的方法也不同，下面分别进行阐述。

1. 如果目标变量是离散型

这里要分 3 种情况：所有预测变量都是离散型，所有预测变量都是连续型，预测变

量有些是离散型、有些是连续型。

1）所有预测变量都是离散型

这里将用到下列表示符号：

X　某个离散的预测变量，有 I 个不同的取值

Y　离散型目标变量，有 J 个不同的取值

N　总的样本数量

N_{ij}　$X=i$ 且 $Y=j$ 的样本数量

$N_{i\cdot}$　$X=i$ 的样本数量，$N_{i\cdot}=\sum_{j=1}^{J}N_{ij}$

$N_{\cdot j}$　$Y=j$ 的样本数量，$N_{\cdot j}=\sum_{i=1}^{I}N_{ij}$

可以用多种方法来度量两个离散型变量之间的独立性或者关联性。

（1）基于皮尔逊 χ^2 的方法

χ^2 检验常用于独立性检验（Test of Independence），用来判断两个离散变量之间是否存在联系，如果不相互关联，就称为独立。在这里，如果预测变量和目标变量之间是相互独立的，那么这个预测变量对目标变量的影响很小，其 importance 指数就一定很小。

例如要根据表 6-1 所示的数据库来检验“地区来源”和“等级”之间的独立性。首先根据这两个变量的取值统计样本数量，并建立交叉表，如表 6-2 所示。

表 6-2　根据两个变量建立交叉表

	一级	二级	三级	合计
甲地区	52	64	24	140
乙地区	60	59	52	171
丙地区	50	65	74	189
合计	162	188	150	500

表中的数字表示样本数量。例如第一个单元中的数字“52”表示地区来源=“甲地区”且等级=“一级”的样本数量为 52。

从表 6-2 可以推出：

地区来源=“甲地区”的概率：P(甲地区)=140/500

等级=“一级”的概率：P(一级)=162/500

那么，如果“地区来源”和“等级”这两个变量是独立的，根据独立性的概率乘法公式，可以计算出第一个单元(甲地区,一级)的期望比例：

P(甲地区,一级)= P(甲地区)×P(一级)=(140/500)×(162/500)=0.09072

0.09072 就是第一个单元中的期望比例，其相应的频数期望值为 0.09072×500=45.36。

以此类推，可以依如下公式计算出每个单元中的频数期望值：$\hat{N}_{ij}=N_{i\cdot}N_{\cdot j}/N$。这样得到一个 $I\times J$（此例为 3×3）的列联表，如表 6-3 所示。

令卡方统计量为：

$X^2=\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{(N_{ij}-\hat{N}_{ij})^2}{\hat{N}_{ij}}$，该值越大，表示变量间的关联度也越高。

表 6-3　3×3 列联表

行	列	N_{ij}	$\hat{N}_{ij}$	$N_{ij}-\hat{N}_{ij}$	$(N_{ij}-\hat{N}_{ij})^2$	$\frac{(N_{ij}-\hat{N}_{ij})^2}{\hat{N}_{ij}}$
1	1	52	45.36	6.64	44.09	0.97
1	2	64	52.64	11.36	129.05	2.45
1	3	24	42.00	–18	324	7.71
2	1	60	55.40	4.60	21.16	0.38
2	2	59	64.30	–5.3	28.09	0.44
2	3	52	51.30	0.7	0.49	0.01
3	1	50	61.24	–11.24	126.34	2.06
3	2	65	71.06	–6.06	36.72	0.52
3	3	74	56.70	17.30	299.29	5.28

该统计量近似服从自由度 $d=(I-1)(J-1)$的卡方分布 χ_d^2 。

基于 χ^2 统计量的 p 值计算为：$p=\text{Prob}(\chi_\alpha^2>X^2)=\alpha$ ，如图 6.1 所示。

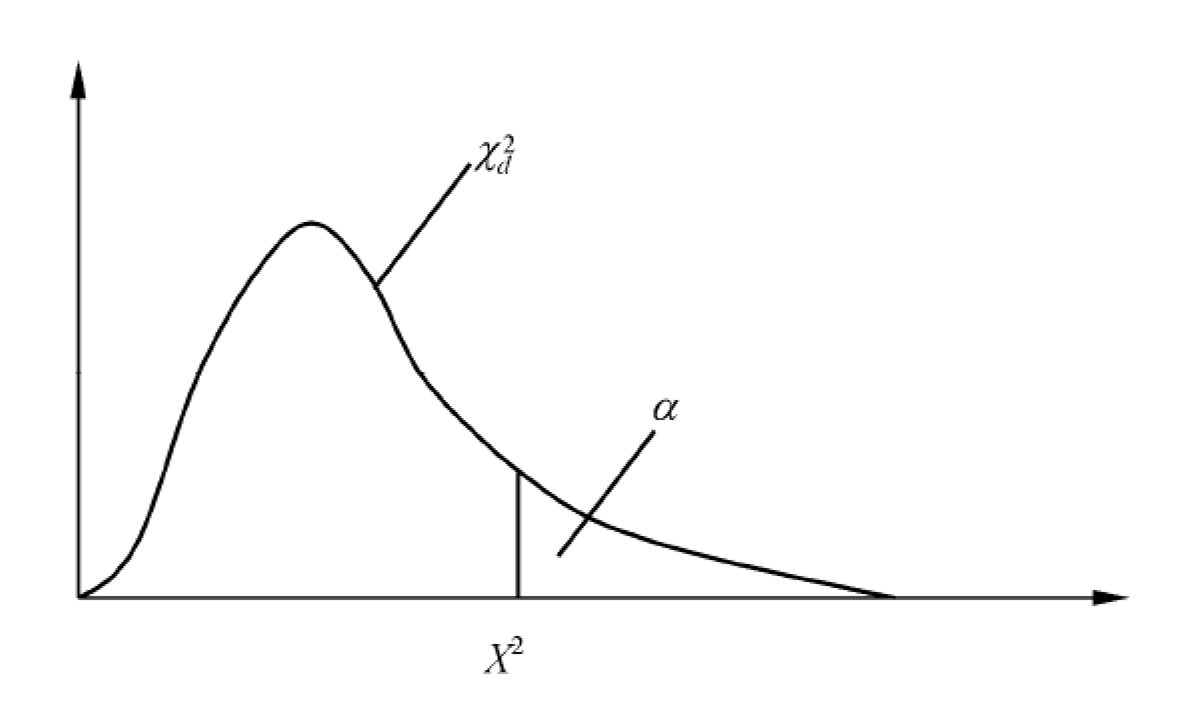


图 6.1　χ_d^2 分布

预测变量 X 的重要程度（也就是 X 与目标变量 Y 之间的关联程度）定义为：

importance=1－Prob($\chi_\alpha^2>X^2$)=1－α 。也就是在求出 X^2 之后，根据 X^2 的值查 χ^2 分布表得出对应的显著水平（p 值），然后即可求出 importance 指数。

在本例中，X^2=19.82，自由度为 d=(3–1)(3–1)=4，查 χ^2 分布表可得 α =0，则 importance=1。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 p 值大小升序排列；

② 若有相同者，按 χ^2 统计量大小降序排列；

③ 若仍有相同者，按自由度 d 的大小升序排列；

④ 若仍有相同者，按在数据文件中的自然顺序排列。

（2）基于似然率 χ^2 的方法

似然率 χ^2 是一种通过检验观测频数与期望频数的比值来分析变量 X 和变量 Y 之间的独立性的方法。同样，每个单元中的频数期望值为 $\hat{N}_{ij}=N_{i.}N_{.j}/N$ 。似然率 χ^2 统计量

定义为

$$G^2=2\sum_{i=1}^{I}\sum_{j=1}^{J}G_{ij}^2\text{，其中，}G_{ij}^2=\begin{cases}N_{ij}\ln\left(\dfrac{N_{ij}}{\hat{N}_{ij}}\right) & N_{ij}>0\\0 & \text{else}\end{cases}$$

该统计量近似服从自由度 $d=(I-1)(J-1)$ 的卡方分布 χ_α^2。

预测变量 X 的重要程度（也就是 X 与目标变量 Y 之间的关联程度）定义为：

importance=1－Prob($\chi_\alpha^2>G^2$)=1－ α 。也就是在求出 G^2 之后，根据 G^2 的值查 χ^2 分布表得出显著水平，然后即可求出 importance 指数。

在本例中，$G^2=2\times10.37=20.74$，自由度为 $d=(3-1)(3-1)=4$，查 χ^2 分布表可得 $\alpha=0$，则 importance=1。

最后，预测变量的排序规则如（1）。

（3）ϕ 系数、C 系数和 V 系数

在皮尔逊 χ^2 的基础上，衍生出了 ϕ 系数和 C 系数，都是利用 χ^2 来计算的相关系数。单位频次的 χ^2 值就构成了 ϕ 相关系数

$$\phi=\sqrt{\frac{X^2}{N}}\text{，这里的 }X^2=\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{(N_{ij}-\hat{N}_{ij})^2}{\hat{N}_{ij}}$$

ϕ 系数适合于分析 2×2 的列联表（变量 X 和变量 Y 各自只有两个取值，$I=J=2$），在这种情况下，如果 $\phi=0$，表示变量 X 和 Y 之间完全不相关；如果 $\phi=1$ 表示变量 X 和 Y 之间完全相关。但是当 I 和 J 均大于 2 时，ϕ 值将随着 I、J 的增大而增大，且没有上限，这时系数之间就缺乏了比较。

为了克服 ϕ 系数的这个缺点，皮尔逊提出了 C 系数

$$C=\sqrt{\frac{X^2}{X^2+N}}\text{，这里的 }X^2=\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{(N_{ij}-\hat{N}_{ij})^2}{\hat{N}_{ij}}$$

当列联表中两个变量 X 和 Y 不相关时，$C=0$；如果两个变量相关，则 C 值随 I 和 J 的变化而变化，但 $0\leqslant C<1$。“$C<1$”这个特点反映出 C 系数的一个缺陷，因为人们在习惯上认为当 X 和 Y 完全相关时，其相关系数应该等于 1。尽管如此，由于 C 系数适用于最低层次的测量尺度，并且对总体的分布形式没有任何要求，因此 C 系数的应用较广。

由克拉默提出的 V 系数（Cramer's V）避免了 ϕ 值无上限及 C 值的上限小于 1 的不足，是一个较为适用的以 χ^2 值为基础的相关系数，V 系数的计算公式为

$$V=\sqrt{\frac{\phi^2}{\min[(I-1),(J-1)]}}=\sqrt{\frac{X^2}{N[\min(I,J)-1]}}$$

当变量 X 与变量 Y 不相关时，$V=0$；当两个变量完全相关时，$V=1$；其他情况下，$0<V<1$。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 V 系数大小降序排列。

② 若有相同者，按 χ^2 统计量大小降序排列。

③ 若仍有相同者，按在数据文件中的自然顺序排列。

（4）λ 系数（Lambda）

λ 系数是一种以减少误差比例（Proportional Reduction In Error）为基础的相关性测量指标。当分析变量 X 和变量 Y 之间的相关程度时，可以通过“不知道 Y 与 X 有关系时，预测 Y 时的全部误差 E_1”和“知道 Y 与 X 有关系时，用 X 去预测 Y 的误差 E_2”的相对差值的大小来度量，这种方法又称为减少误差比例法：

$$\lambda = \frac{E_1 - E_2}{E_1}$$

其中，E_1 为不知道 Y 与 X 有关系时，预测 Y 时的全部误差；E_2 为知道 Y 与 X 有关系时，用 X 去预测 Y 的全部误差。

可见 $E_1 - E_2$ 表示知道 Y 与 X 有关系后，预测 Y 所减少的误差。而 $\frac{E_1 - E_2}{E_1}$ 则表示所减少的相对误差。$\frac{E_1 - E_2}{E_1}$ 越大，则表示 Y 和 X 的关系越密切，相关程度越高。

当 Y 与 X 完全不相关时，$E_1 = E_2$，此时 $\lambda = 0$。

当 Y 与 X 完全相关时，$E_2 = 0$，此时 $\lambda = 1$。

其他情况下，λ 介于 0 和 1 之间。

E_1 的定义为：当不知道 Y 与 X 有关系时，如果来预测 Y 的值，唯一可以参考的就是 Y 本身的分布，即关于 Y 的边缘分布。可以直接用其边缘分布中的众数去猜测 Y 的值（一组数据中出现次数最多的那个数据，叫做这组数据的众数）。显然，用这个值来猜测 Y，比用其他值来猜测 Y，猜中的概率都要高。众数所对应的频次为 $\max\limits_j(N_{.j})$，也就是 j 个频次中最大的那一个，那么这样预测 Y 的误差为

$$E_1 = N - \max_j(N_{.j})$$

E_2 的定义为：当知道 Y 与 X 有关系后，如果要预测 Y 的值，则根据 X 的值所对应的 Y 的众数去猜测 Y 值。也就是用条件分布中的众数去预测 Y 的值，这样猜中的频次最多，误差最小。

设 $X=i$ 时，条件分布中众数的频次为 $\max\limits_j(N_{ij})$，也就是 X=i 的前提下，j 个频次中最大的那一个，在这时预测 Y 产生的误差为

$$E_2 = N - \sum_i \max_j(N_{ij})$$

因此，

$$\lambda(Y|X) = \frac{E_1 - E_2}{E_1} = \frac{\sum_i \max_j(N_{ij}) - \max_j(N_{.j})}{N - \max_j(N_{.j})}$$

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 λ 系数大小降序排列。

② 若有相同者，按变量的取值个数 I 的大小升序排列。

③ 若仍有相同者，按在数据文件中的自然顺序排列。

2）如果所有预测变量都是连续型

当分析一个连续变量与一个离散变量之间的关系时，可以采用单因素方差分析的方法（一元方差分析，one-way Analysis Of Variance，one-way ANOVA），这里采用的是基于 F 统计量的 p 值检验方法。

这里将用到下列表示符号：

N_j　目标变量 $Y=j$ 的样本数量

$\bar{x}_j$　对于所有 $Y=j$ 的样本，预测变量 X 的样本均值

$$\bar{x}_j = \frac{1}{N_j}\sum_{i=1}^{I} x_{ij}$$

s_j^2　对于所有 $Y=j$ 的样本，预测变量 X 的样本方差

$$s_j^2 = \frac{\sum_{i=1}^{N_j}(x_{ij}-\bar{x}_j)^2}{N_j-1}$$

$\bar{\bar{x}}$　预测变量 X 的总体均值

$$\bar{\bar{x}} = \frac{1}{N}\sum_{j=1}^{J} N_j \bar{x}_j$$

各观测值 x_{ij} 对本组（所有 $Y=j$ 的样本）均值 $\bar{x}_j$ 的偏差平方和的总和记为

$$RSS = \sum_{i=1}^{N_j}(x_{ij}-\bar{x}_j)^2 = s_j^2(N_j-1)$$

观测值的组均值 $\bar{x}_j$ 对总体均值 $\bar{\bar{x}}$ 的偏差平方和记为：

$$BSS = \sum_{j=1}^{J} N_j(\bar{x}_j-\bar{\bar{x}})^2$$

这里，BSS 反映了各组或各类样本之间的差异程度，它是由于 Y 的取值不同所引起的。而 RSS 则是由其他未知因素所引起的误差。那么，统计量

$$F = \frac{BSS/(J-1)}{RSS/(N-J)} = \frac{\sum_{j=1}^{J} N_j(\bar{x}_j-\bar{\bar{x}})^2/(J-1)}{s_j^2(N_j-1)/(N-J)}$$

服从分子自由度 $K_1=J-1$、分母自由度 $K_2=N-J$ 的 F 分布，即 $F\sim F(J-1, N-J)$。

基于 F 统计量的 p 值计算为：$p=\text{Prob}\{F(J-1,N-J)>F\}=\alpha$，如图 6.2 所示。

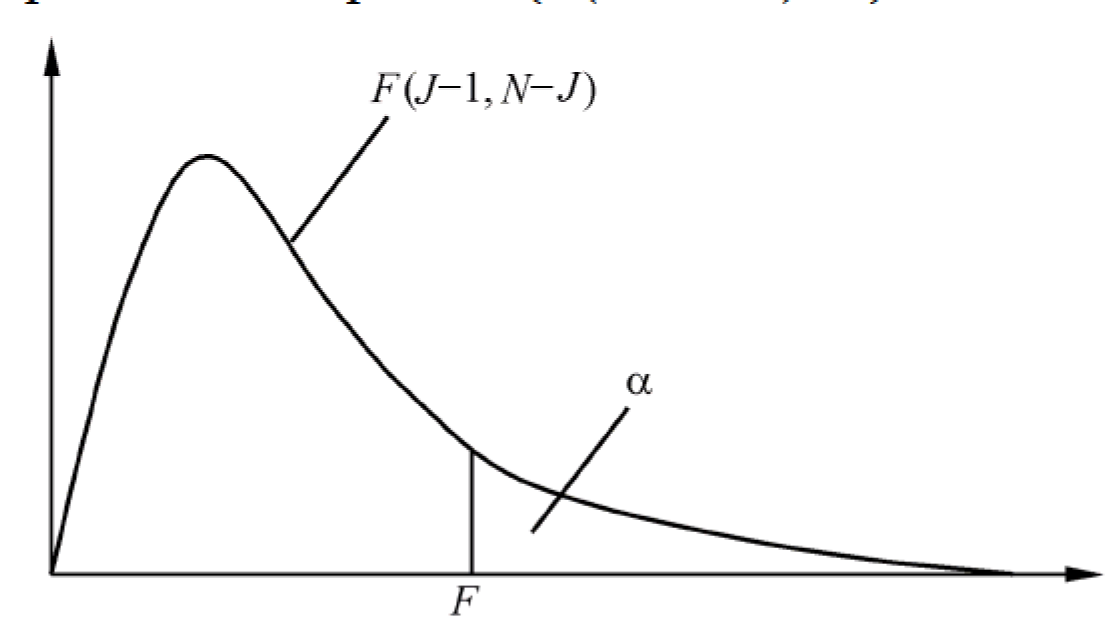


图 6.2　$F(J-1,N-J)$分布

预测变量 X 的重要程度定义为：

importance=1–Prob{$F(J-1,N-J)>F$}=1–α。也就是在求出 F 之后，根据 F 的值查 F 分布表得出对应的显著水平（p 值），然后即可求出 importance 指数。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 p 值大小升序排列。

② 若有相同者，按 F 统计量的大小降序排列。

③ 若有相同者，按样本量 N 的大小降序排列。

④ 若仍有相同者，按在数据文件中的自然顺序排列。

3）预测变量有些是离散型，有些是连续型

如果预测变量有些是离散型，有些是连续型，那么对那些连续型预测变量，采用基于 F 统计量的 p 值来计算 importance 指数；对那些离散型变量，则用基于皮尔逊卡方的 p 值（或者基于似然率卡方的 p 值）来计算 importance 指数。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 p 值大小升序排列。

② 若有相同者，先将它们分为“连续型”和“离散型”两类，分别用上面阐述的“所有变量是连续型”和“所有变量是离散型”两种情况下的排序规则来排序。最后，比较排序后这两类中各自的第一个变量在数据文件中的顺序，如果“连续型”类中的第一个变量在数据文件中的顺序在前，那么“连续型”类中的变量就排在“离散型”类中变量之前。反之，如果“离散型”类中的第一个变量在数据文件中的顺序在前，那么“离散型”类中的变量就排在“连续型”类中变量之前。

2．如果目标变量是连续型

这里也要分 3 种情况：所有预测变量都是离散型，所有预测变量都是连续型，预测变量有些是离散型、有些是连续型。

（1）所有预测变量都是离散型

如果所有预测变量都是离散型，且目标变量是连续型，同样可以采用基于 F 统计量的 p 值检验方法。

这里将用到下列表示符号：

X　有 I 个不同取值的某个离散型预测变量

Y　连续型的目标变量。y_{ij} 表示第 j 个 $X=i$ 的样本的目标变量 Y 的值

N_i　$X=i$ 的样本的数量

$\overline{y}_i$　对于所有 $X=i$ 的样本，Y 的样本均值

$$\overline{y}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} y_{ij}$$

$s(y)_i^2$　对于所有 $X=i$ 的样本，Y 的样本方差

$$s(y)_i^2 = \frac{\sum_{j=1}^{N_i}(y_{ij}-\overline{y}_j)^2}{N_i - 1}$$

$\bar{\bar{y}}$ Y的总体均值

$$\bar{\bar{y}}=\frac{1}{N}\sum_{i=1}^{I}N_i\bar{y}_j$$

那么构造统计量 F

$$F=\frac{\sum_{i=1}^{I}N_i(\bar{y}_j-\bar{\bar{y}})^2/(I-1)}{s(y)_i^2(N_i-1)/(N-I)}$$

服从分子自由度 $K_1=I-1$、分母自由度 $K_2=N-I$ 的 F 分布，即 $F\sim F(I-1, N-I)$。

基于 F 统计量的 p 值计算为 $p=\text{Prob}\{F(J-1,N-J)>F\}=\alpha$。

预测变量 X 的重要程度定义为

$\text{importance}=1-\text{Prob}\{F(I-1,N-I)>F\}=1-\alpha$。也就是在求出 F 之后，根据 F 的值查 F 分布表得出对应的显著水平（p 值），然后即可求出 importance 指数。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 p 值大小升序排列；

② 若有相同者，按 F 统计量的大小降序排列；

③ 若有相同者，按样本量 N 的大小降序排列；

④ 若仍有相同者，按在数据文件中的自然顺序排列。

（2）如果所有预测变量都是连续型

这里将用到下列表示符号：

X 某个连续型预测变量

Y 连续型目标变量

$\bar{x}=\sum_{i=1}^{N}\frac{x_i}{N}$ 预测变量 X 的样本均值

$\bar{y}=\sum_{i=1}^{N}\frac{y_i}{N}$ 目标变量 Y 的样本均值

$s(x)^2=\frac{\sum_{i=1}^{N}(x_i-\bar{x})}{N-1}$ 预测变量 X 的样本方差

$s(y)^2=\frac{\sum_{i=1}^{N}(y_i-\bar{y})}{N-1}$ 目标变量 Y 的样本方差

从概率论的知识可知，两个正态随机变量 X、Y 之间的关系由协方差来刻度

$$\text{cov}(X,Y)=E(X-EX)(Y-EY)$$

变量 X、Y 之间的线性关系的近似程度用相关系数 ρ 来刻度

$$\rho=\frac{E(X-EX)(Y-EY)}{\sqrt{D(X)D(Y)}}=\frac{E(X-EX)(Y-EY)}{E(X-EX)^2(Y-EY)^2}$$

一般来说，$|\rho|$ 越接近于 1，X 和 Y 之间越近似地有线性关系。由于在统计推论中把随机变量 X、Y 看成两个总体，因此也把 ρ 称为总体相关系数。由概率论还知，$|\rho|\leqslant 1$。且当 $\rho=1$ 时，以概率 1 存在 a 和 $b(b>0)$，使得 $Y=a+bX$ 成立；当 $\rho=-1$ 时，以概率 1

存在 a 和 $b(b<0)$，使得 $Y=a+bX$ 成立；当 $\rho=0$ 时，X 和 Y 不相关。所以，两个正态随机变量之间的线性联系程度完全可以用总体相关系数来刻画。然而当总体中包含的样本数量非常庞大时，事实上是无法计算出 ρ 的。因此必须能够根据数量有限的样本，来计算 ρ 的估计值，并根据这个估计值来刻画变量之间的关系。

假设从总体中随机抽取 N 个样本点(x_1, y_1)，(x_2, y_2)，…，(x_N, y_N)，那么样本的均值为：$\overline{x}=\sum_{i=1}^{N}\frac{x_i}{N}$，$\overline{y}=\sum_{i=1}^{N}\frac{y_i}{N}$。

在 XOY 平面上作出点(x_i, y_i)，$i=1,2,\cdots,N$，得到如图 6.3 所示的散点图。

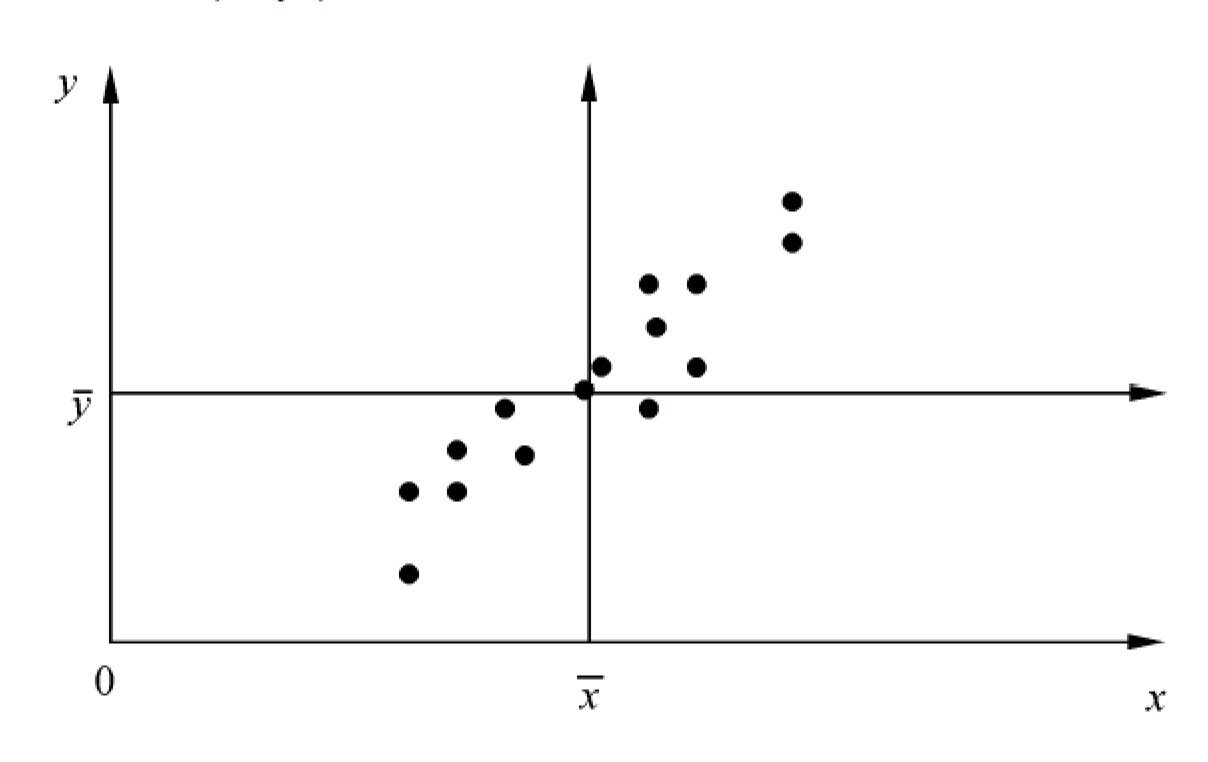


图 6.3　样本散点图

将坐标轴平移至$(\overline{x},\overline{y})$，在这个新坐标系下，如果点$(x_i, y_i)$落在新坐标系的第 I 和第 III象限，则有 $(x_i-\overline{x})(y_i-\overline{y})>0$；如果点$(x_i, y_i)$落在新坐标系的第 II 和第 IV 象限，则有 $(x_i-\overline{x})(y_i-\overline{y})<0$。

不难发现，如果所有的样本点基本上都落在第 I 和第 III 象限，则 $\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})>0$，这时称 X 和 Y 正线性相关；如果所有的样本点基本上都落在第 II 和第 IV 象限，则 $\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})<0$，这时称 X 和 Y 负线性相关；如果 $\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})=0$，则称 X 和 Y 之间不存在线性关系，此时，通常样本点会比较均匀地散落在 4 个象限中。

因此，可以很自然地想到用数值 $\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})$ 来衡量 X 和 Y 之间线性相关程度强弱的度量。但还有两个问题，其一，变量 X 和变量 Y 的计量单位不相同；其二，相关程度的强弱必须相对于一个可供对比的标准，理想的标准是把相关关系的度量控制在[–1,1]里。为此，我们将变量标准化，然后再求其乘积的平均，得到

$$r=\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{x_i-\overline{x}}{s(x)^2}\right)\left(\frac{y_i-\overline{y}}{s(y)^2}\right)=\frac{\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\overline{x})^2\sum_{i=1}^{N}(y_i-\overline{y})^2}}$$

r 有以下特征：

① $|r|\leqslant 1$。

② 当 $|r|=1$ 时，X 和 Y 完全线性相关。其中 $r=+1$ 时称为完全正相关；$r=-1$ 时称为完全负相关。

③ 当 $r=0$ 时，X 和 Y 不相关。

④ 在绝大多数情况下，$-1<r<1$，这时 X 和 Y 存在一定的线性关系。

确定统计量 t，

$$t=r\sqrt{\frac{N-2}{1-r^2}}$$，该统计量服从自由度为 $N-2$ 的 t 分布。

基于 t 统计量的 p 值计算为

$$p=\begin{cases}0 & r^2=1\\ 2p(t>|t_\alpha|)=2\alpha & \text{else}\end{cases}$$

如图 6.4 所示。

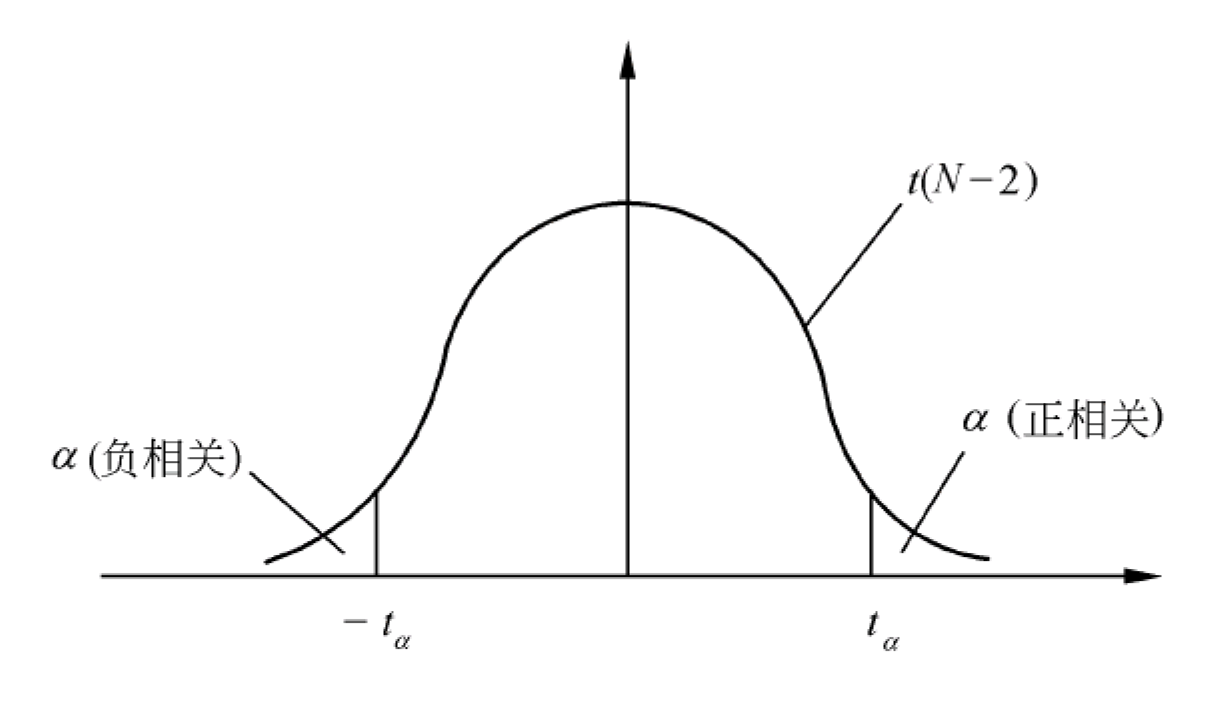


图 6.4　$t(N-2)$分布

然后，importance=1−p。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 p 值大小升序排列；

② 若有相同者，按 r^2 的大小降序排列；

③ 若有相同者，按样本量 N 的大小降序排列；

④ 若仍有相同者，按在数据文件中的自然顺序排列。

（3）预测变量有些是离散型，有些是连续型

如果预测变量有些是离散型，有些是连续型。对于连续型预测变量，采用上述基于 t 统计量的 p 值检验方法；对于离散型预测变量，则用基于 F 统计量的 p 值检验方法。

最后，根据以下规则对预测变量排序，并从“1”开始生成秩。

① 根据 p 值大小升序排列。

② 若有相同者，先将它们分为“连续型”和“离散型”两类，分别用上面阐述的“所有变量是连续型”和“所有变量是离散型”两种情况下的排序规则来排序。最后，比较排序后这两类中各自的第一个变量在数据文件中的顺序，如果“连续型”类中的第一个变量在数据文件中的顺序在前，那么“连续型”类中的变量就排在“离散型”类中变量之前。反之，如果“离散型”类中的第一个变量在数据文件中的顺序在前，那么“离散

型”类中的变量就排在“连续型”类中变量之前。

6.1.4 选择

在本阶段，根据前面的计算结果，来选择参与建模的预测变量。根据用户的不同设置，可以有多种不同的选择规则。

1．根据预测变量的类别来选择

特征选择算法最终根据各预测变量的 importance 指数将它们划分为 3 类：Important（重要）、Marginal（一般重要）、Unimportant（不重要）。其中每一类都可以由用户自行设定 importance 指数的阈值。例如 Important 类的 importance 指数必须满足 0.93≤importance≤1，Marginal 类的 importance 指数必须满足 0.9＜importance≤0.93，其他属于 Unimportant 类。用户可以自行指定选择 Important 类的预测变量参与建模，或者选择 Important 类和 Marginal 类的预测变量参与建模。

2．选择 importance 指数最大的前 k 个预测变量参与建模

可以用下述方法来确定 k 的数值：

设 L_0 表示全部待选预测变量的个数，那么 k 可以用以下公式来估计：

$$k=[\min(\max(30,2\sqrt{L_0}))]$$

其中“[ ]”符号的含义是：$[x]$表示最接近于 x 的那个整数。

根据这一公式，当 L_0 为不同的取值时，k 的取值情况如表 6-4 所示。

表 6-4　根据 L_0 确定 k

L_0	k	k/L_0（%）
10	10	100.00
15	15	100.00
20	20	100.00
25	25	100.00
30	30	100.00
40	30	75.00
50	30	60.00
60	30	50.00
100	30	30.00
500	45	9.00
1000	63	6.30
1500	77	5.13
2000	89	4.45
5000	141	2.82
10000	200	2.00
20000	283	1.42
50000	447	0.89

3. 根据 importance 指数

用户可直接指定 importance 指数阈值，大于阈值的预测变量被选择出来参与建模。

4. 人工选择

参考得到的 importance 指数，用户根据实际需要和领域知识自行选择。

6.1.5 在 Clementine 中应用特征选择

本节通过一个实例来演示在 Clementine 中特征选择节点的用法。

事务数据集为 Clementine 自带的 customer_dbase 数据集，存放在 Clementine 安装目录下的 Demos 文件夹中（...\Clementine12\Demos\customer_dbase.sav），包含了 132 个字段，5000 条记录。假设现在要根据数据集中的 response_01 字段来建立分类模型，由于字段数量很多，可以先用特征选择算法，筛选出那些对 response_01 字段影响较大的字段来参与建模，而忽略那些对该字段影响不大的字段。

完整的数据流如图 6.5 所示。

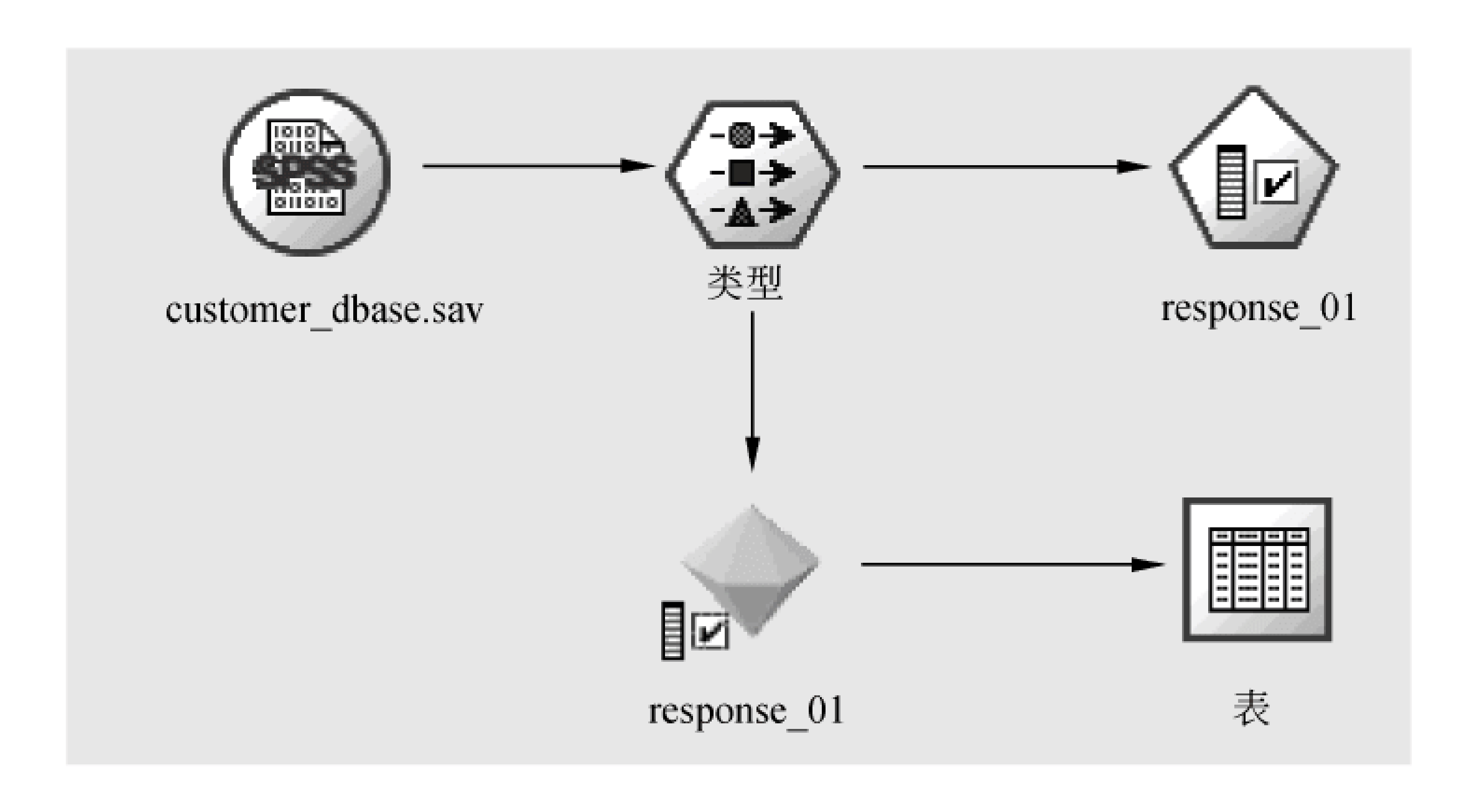


图 6.5　特征选择数据流

首先，将“数据源”中的“SPSS 文件”节点添加到数据流区域，并将 customer_dbase.sav 文件加载到该节点。

向数据流中添加“类型”节点，并建立从 customer_dbase.sav 节点到“类型”节点的连接。打开“类型”节点的编辑窗口，把不需要分析的字段的方向设置为“无”（这里把 custid 的方向设置为“无”），把目标字段 response_01 的方向设置为“输出”，其他字段的方向为“输入”，如图 6.6 所示。这里，输入字段就是预测变量，输出字段就是目标变量。然后单击“读取值”按钮。

向数据流中添加“特征选择”节点，建立从“类型”节点到“特征选择”节点的连接，然后对“特征选择”节点进行设置，在节点编辑窗口的“模型”标签下，设置如图 6.7 所示。



图 6.6　设置“类型”节点

response_01

模型名称: 自动 自定义

使用分区数据

屏幕字段:

缺失值的最大百分比	70.0	(所有字段)
单个类别中记录的最大百分比	90.0	(类别)
作为记录百分比的最大类别数	95.0	(类别)
最小变异系数	0.1	(范围)
最小标准差	0.0	(范围)

字段　模型　选项　注解

确定　执行　取消　应用　重置

图 6.7　设置筛选阈值

这里有 5 个筛选字段的标准以及相应的阈值设置微调框。用户可勾选一个或多个筛选标准并设置相应的阈值，满足勾选标准及阈值的字段将被屏蔽。这 5 个阈值实际上就是第 6.1.2 节中阐述的参数 m_1、m_2、m_3、m_4、m_5：

缺失值的最大百分比（Maximum Percentage of Missing Values）：m_1

单个类别中记录的最大百分比（Maximum Percentage of Records In A Single Category）：m_2

作为记录百分比的最大类别数（Maximum Number of Categories As A Percentage of Records）：m_5

最小变异系数（Minimum Coefficient of Variation）：m_4

最小标准差（Minimum Standard Deviation）：m_3

在“选项”标签下的设置如图 6.8 所示。



图 6.8　设置字段选择标准和 *p* 值计算方法

这里提供了 3 个选择字段的标准，用于选择那些对目标变量影响较大的预测变量。

所有排列的字段（All Fields Ranked）：通过设置相应的阈值，将预测变量划分为“重要”、“一般重要”和“不重要”3 类，并通过勾选某类来选择该类的预测变量。

字段总数（Top Number of Fields）：选择 importance 指数最大的前 *k* 个预测变量。

重要性大于（Importance Greater Than）：选择 importance 指数大于指定值的预测变量。

对于分类预测变量（离散型变量），选择计算 *p* 值的方法。从 4 种方法中选择其一，这 4 种方法的详细描述见第 6.1.3 节。

以上设置结束后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示生成的特征选择模型（Generated Feature Selection Model）“response_01”。右击该模型，在快捷菜单中选择“浏览”命令，即可打开模型对话框，如图 6.9 所示。

该窗口由两个子窗口组成。在下半部分的子窗口中，显示了在筛选阶段屏蔽的 9 个字段。在上半部分的子窗口中，显示了按秩排列的所有预测变量的字段名、类型、重要性和 *importance* 指数值，并在那些满足选择标准的预测变量前面打勾，表示已选择，这里选择了 34 个预测变量。当然，用户还可以根据实际情况勾选其他自认为重要的变量，包括下面被屏蔽的 9 个字段中的一个或多个。

将生成的 response_01 节点添加到数据流中，并建立由“类型”节点到该节点的连接，即可用该模型对原始数据集进行筛选，因此，在 response_01 节点之后再添加某个分类建模节点，即可对筛选之后的数据集进行分类建模了。

可以通过在生成的 response_01 节点后添加“表”节点来浏览经过 response_01 节点过滤之后的结果。

添加“表”节点，并建立由 response_01 节点到“表”节点的连接，并执行“表”节

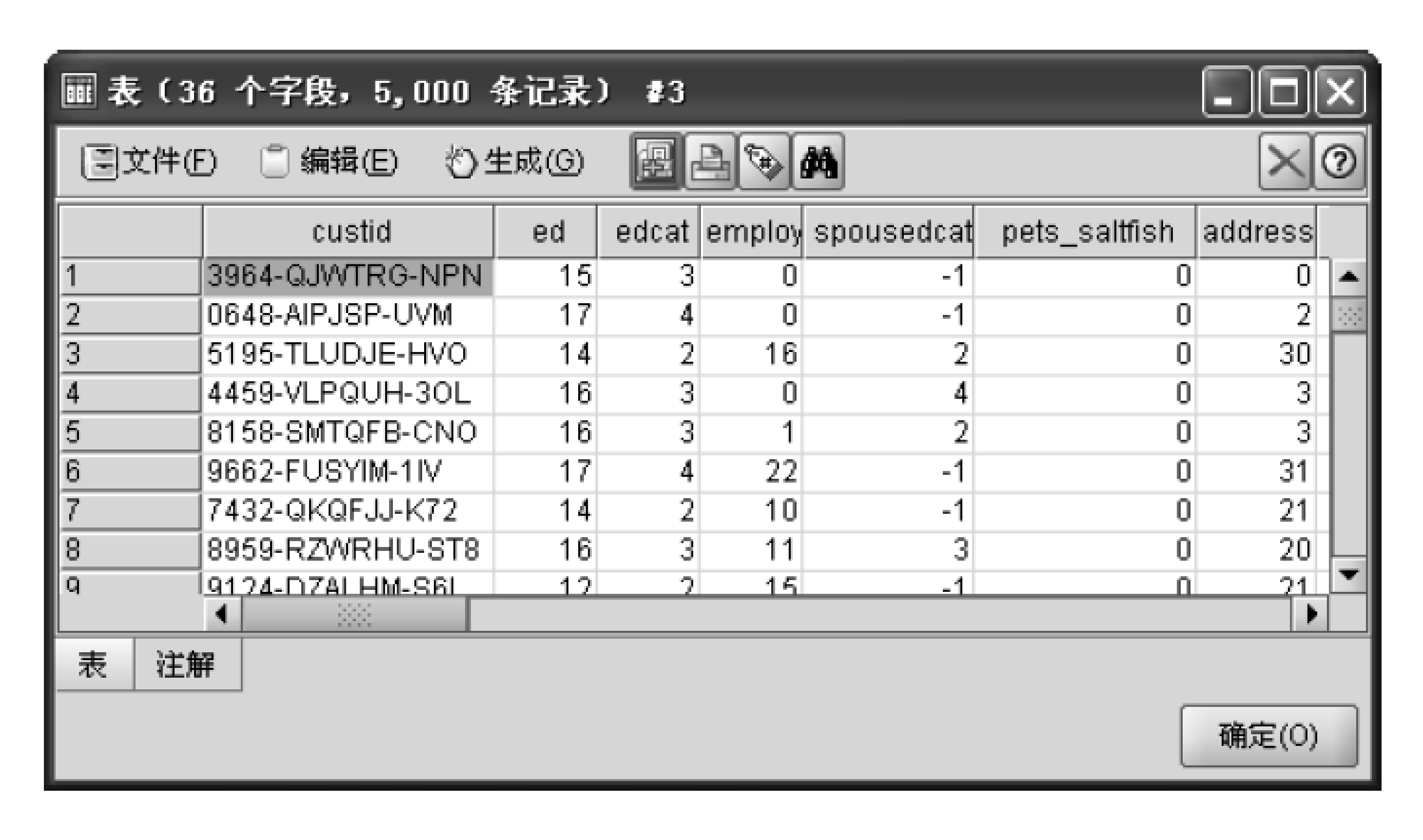
点，结果如图 6.10 所示。



等级	字段	类型	重要性	值
1	ed	范围	重要	1.0
2	ownpc	集	重要	1.0
3	edcat	有序集合	重要	1.0
4	internet	集	重要	1.0
5	equip	集	重要	1.0
6	owngame	集	重要	1.0
7	equipmon	范围	重要	1.0
8	confer	集	重要	1.0
9	ebill	集	重要	1.0
10	callwait	集	重要	1.0
11	forward	集	重要	1.0
12	tollmon	范围	重要	1.0
13	multline	集	重要	1.0
14	ownipod	集	重要	1.0
15	callid	集	重要	1.0
16	equipten	范围	重要	1.0
17	tollfree	集	重要	1.0
18	tollten	范围	重要	1.0

字段	类型	原因
ownvcr	集	单个类别过大
owntv	集	单个类别过大
owndvd	集	单个类别过大
owncd	集	单个类别过大
lnwireten	范围	缺失值过多
lnwiremon	范围	缺失值过多
lnequipm...	范围	变异系数低于阈值
commute...	集	单个类别过大

图 6.9　特征选择结果



	custid	ed	edcat	employ	spousedcat	pets_saltfish	address
1	3964-QJWTRG-NPN	15	3	0	-1	0	0
2	0648-AIPJSP-UVM	17	4	0	-1	0	2
3	5195-TLUDJE-HVO	14	2	16	2	0	30
4	4459-VLPQUH-3OL	16	3	0	4	0	3
5	8158-SMTQFB-CNO	16	3	1	2	0	3
6	9662-FUSYIM-1IV	17	4	22	-1	0	31
7	7432-QKQFJJ-K72	14	2	10	-1	0	21
8	8959-RZWRHU-ST8	16	3	11	3	0	20

图 6.10　特征选择后的数据集

可以看到，该数据集共包含 36 个字段，其中 34 个字段是特征选择的结果，另外两个字段是 custid 和 response_01。因为在设置类型节点时，custid 字段的方向被设置为“无”，即不分析它与目标字段间的关系，所以该字段得以保留。而 response_01 是目标变量，当然被保留下来，所以共有 36 个字段。

6.2 异常检测

6.2.1 异常数据挖掘概述

异常数据挖掘，又称为离群点分析或者孤立点挖掘。在人们对数据进行分析处理的过程中，经常会遇到少量这样的数据，它们与数据的一般模式不一致，或者说与大多数样本相比有些异常。我们把这种数据称为异常数据（Outlier Data）。对异常数据的分析处理在某些领域很有应用价值，例如在网络安全领域，可以利用异常数据挖掘来分析网络中的异常行为；在金融领域，异常数据的识别可以发现信用卡的欺诈交易、股市的操控行为、会计信息的虚报造价、洗钱活动、欺诈贷款等。

异常数据挖掘涉及两个基本的问题。其一，在对一个给定的数据集进行分析之前，必须事先约定满足什么条件的数据才是异常数据，也就是异常数据的定义问题。其二，用什么方法来从给定的数据集中将异常数据提取出来。

1. 异常数据的定义

对于异常数据，Hawkins 的定义是，“异常就是在数据采集中与众不同的数据，使人怀疑这些数据并非随机偏差，而是产生于完全不同的机制”。这个定义说明，异常数据是少量的、与众不同的，与大多数数据相比是有“偏差”的，而且产生这种“偏差”的原因不是随机的（例如录入错误），而是有其更深层次的必然原因，它产生于完全不同的机制。

为了从数据集中识别异常数据，就必须有一个明确的标准。这就需要找到数据的内在规律，在一个可接受的误差范围之内，满足内在规律的数据就是正常数据，而不满足内在规律的数据就是异常数据。这种数据间的内在规律可以根据数据本身的特点从位置关系、函数关系、规则关系、序列关系等方面来考量。

（1）位置关系

位置关系是数据之间最常见的一种关系。大多数正常数据由于具有很大的相似性而符合一个共同的模式，在空间上表现出聚拢在一起的趋势，“团结”在一个或者多个“核心”周围。而那些异常数据则表现得很“离群”，它们总是离所有的“核心”都很远，如图 6.11 所示。

（2）函数关系

函数关系也是一种常见的数据关系，即大多数数据都符合某个函数模型，因此数据点大多分布在函数曲线附近，而那些异常数据则距离曲线较远，如图 6.12 所示。

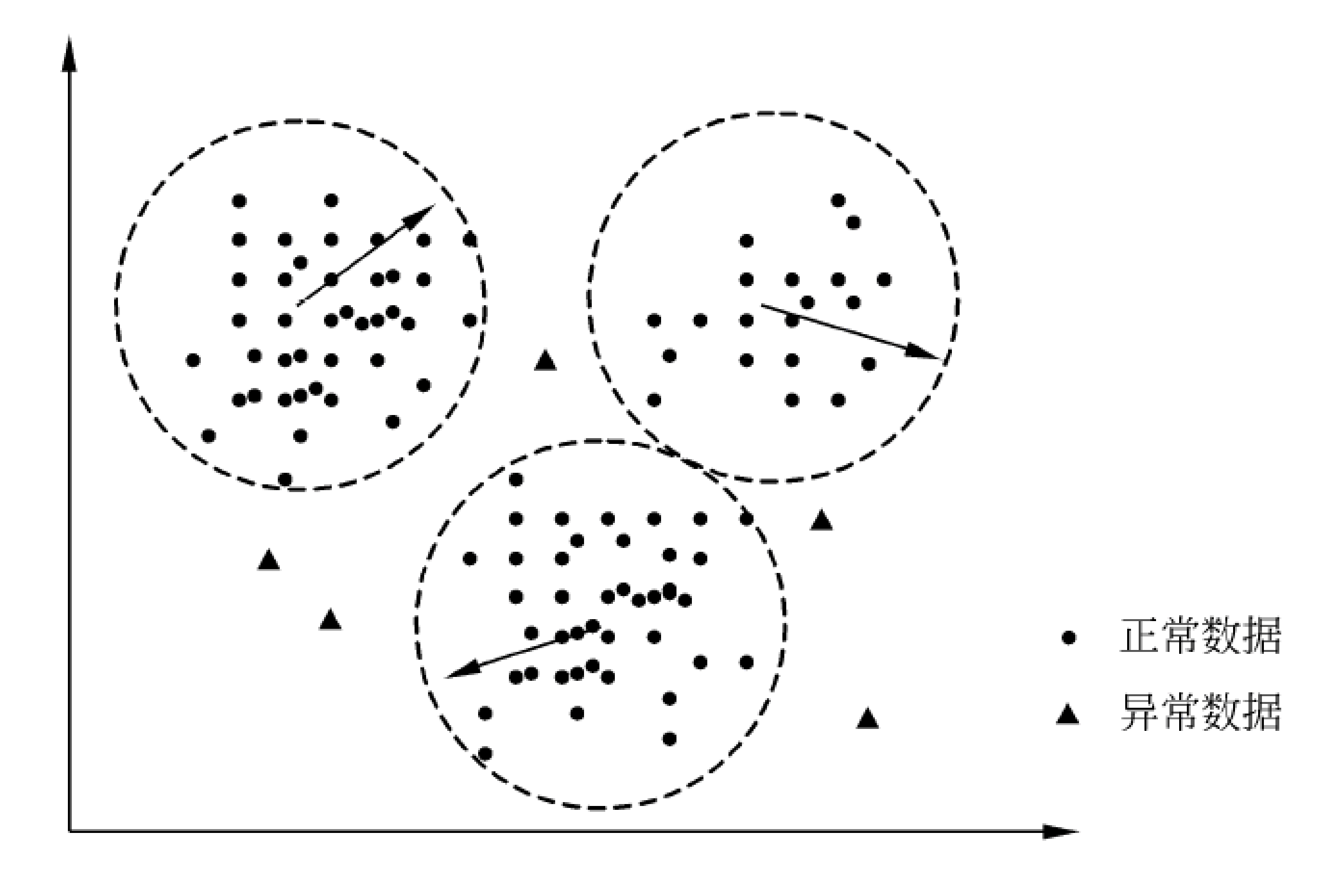


图 6.11 数据之间的位置关系

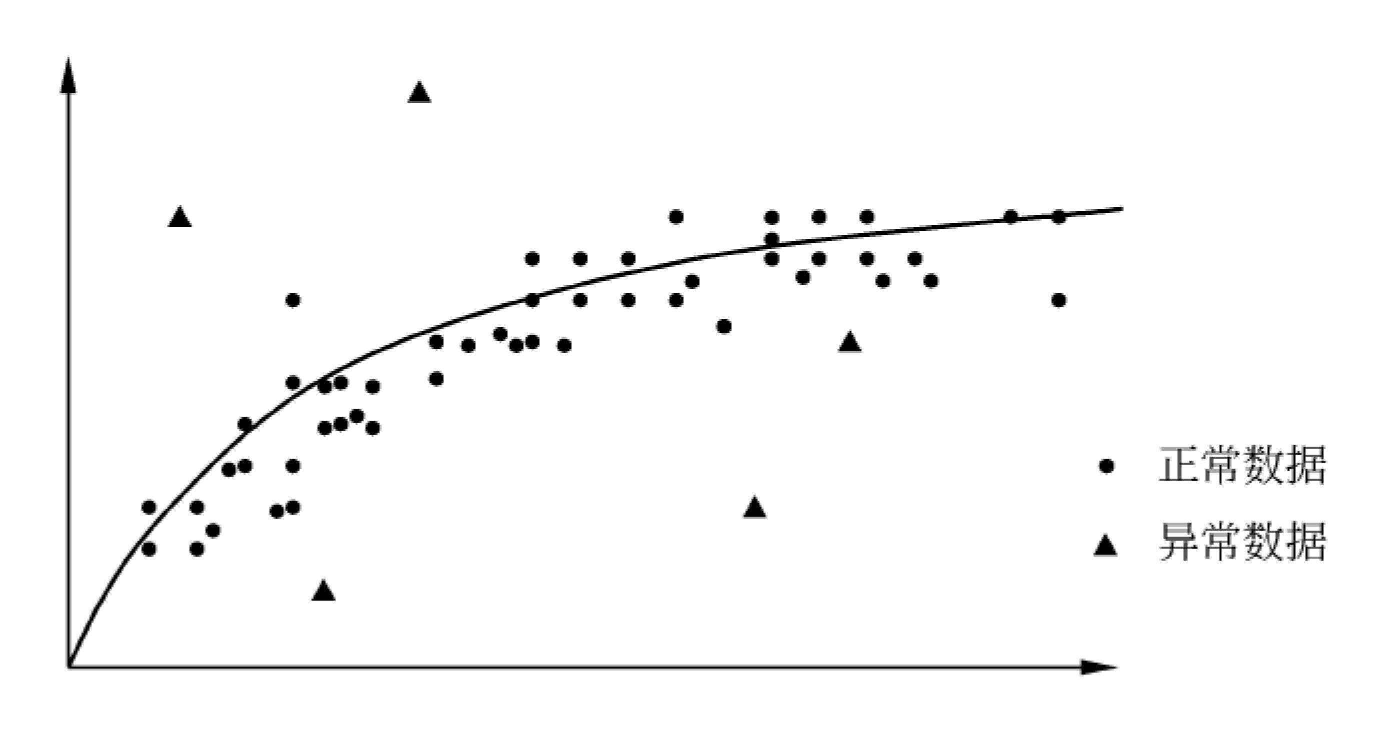


图 6.12 数据之间的函数关系

（3）规则关系

如果数据集中某些数据符合某个规则的条件，则称这些数据具有规则关系。具有同一规则关系的正常数据一般会使该规则的结论成立，而如果某个数据具有该规则关系但不能使规则的结论成立，那么它就是异常数据。规则的一般形式为

$$A_1 \& A_2 \cdots \& A_n \rightarrow C$$

这里 A_i 和 C 都是对数据特征的描述。

（4）序列关系

序列关系是指数据集中的某些数据满足某种序列模式，而那些在相同条件下不满足序列模式的数据就是异常数据。

2. 异常挖掘方法

异常挖掘方法是数据挖掘的一个重要研究方向。许多研究人员从不同的角度以及不同的应用领域出发，提出了不同类型的异常以及挖掘方法。从技术原理来看，这些方法可以划分为基于统计的方法、基于距离的方法、基于密度的方法、基于偏离的方法和基于聚类的方法。

（1）基于统计的方法

基于统计的方法的基本思想是根据数据集的特性事先假定一个数据分布的概率模

型，然后根据模型的不一致性来确定异常。首先对给定的数据集假设一个概率分布模型（例如正态分布），然后在某个显著性水平上，确定数据的拒绝域和接受域，拒绝域是数据出现概率很小的区间，如果数据落在此区域，则判定为异常数据。

统计方法的优点是很明显的。它建立在成熟的统计学理论基础之上，只要给定概率模型，其统计方法非常有效，异常点的含义也非常明确。

但存在的问题是，在许多情况下，我们并不知道数据的分布，而且现实数据也往往不符合任何一种理想状态的数学分布，这样就对后期的异常挖掘产生了很大的困难。另一方面基于统计的方法比较适合于低维空间的异常挖掘，而实际的数据大多都是高维空间的数据，在这种情况下，事先估算数据的分布是很困难的。

（2）基于距离的方法

基于距离的方法主要基于数据点间的距离来发现异常点，由于它具有比较明显的几何解释，是当前使用最为普遍的方法。

基于距离的方法的基本思想是以距离的大小来检测小模式，异常点被认为是那些没有足够多邻居的点。它可以描述为在数据集 N 中，至少有 P 个对象和对象 O 的距离大于 d，则对象 O 是一个带参数 P 和 d 的基于距离的异常点。

基于距离的检测方法的优势在于，它不需要事先了解数据集本身的特性，也与领域无关。但是问题在于对参数 P 和 d 估计的困难性。不同的 P 和 d 参数的确定会对结果带来很大的影响。

（3）基于密度的方法

基于距离的方法对全局各个聚类的数据提出了统一的 P 和 d 的参数，但是如果各个聚类本身的密度存在不同，则基于距离的方法会出现问题，因此提出了基于密度模型的局部异常点挖掘算法。

在这种情况下，数据是否异常不仅仅取决于它与周围数据的距离大小，而且与邻域内的密度状况有关，一个邻域内的密度可以用包含固定数据点个数的邻域半径或者指定半径邻域中包含的数据点数目来描述。

Breunig 和 Kriegel 提出了用局部异常因子（Local Outlier Factor，LOF）来度量对象异常程度的方法。只要一个对象的 LOF 远大于 1，它可能就是一个异常点。簇内靠近核心点的对象的 LOF 接近于 1，处于簇的边缘或是簇的外面的对象的 LOF 相对较大，这样便能检测到局部异常点，更贴近于实际的数据集的特性。这种传统的局部异常点的挖掘算法的主要问题在于局部范围的参数 Minpts 值存在选择上的困难，可以运用多粒度偏差因子代替 Minpts 来评价，这样便能得到比较好的解决方案。

（4）基于偏离的方法

基于偏离的方法的基本思想是通过检查一组对象的主要特征来确定异常点，如果一个对象的特征与给定的“描述”过分地“偏离”，则该对象被认为是异常点。

现有的基于偏离的方法主要有序列异常技术和 OLAP 数据立方体方法。

序列异常技术的核心是要构造一个相异度函数（Dissimilarity Function），对于一个包含了许多样本的数据集，如果样本间的相似度较高，相异度函数的值就比较小，反之，如果样本间的相异度越大，相异度函数的值就越大（例如，方差就是一个满足这种要求的函数）。那么，如果将一个样本从数据集中剔除后，使得该数据集的相异度大大减小，

就可以判定这个样本是异常数据。

OLAP 数据立方体方法利用在大规模的多维数据中采用数据立方体确定反常区域，如果一个立方体的单元值显著地不同于根据统计模型得到的期望值，则该单元值被认为是一个异常点。

（5）基于聚类的方法

基于聚类的方法的基本思想是将异常挖掘的过程转换成聚类的过程。首先将数据集利用已经成熟的模型进行聚类分析，将数据集划分为多个簇，然后选择那些离簇的质心较远的样本作为异常点。

6.2.2 异常检测算法

本小节介绍的是 Clementine 中实现异常检测的算法，该算法在 Clementine 中是由异常节点来执行的。

这是一种结合了“基于距离”和“基于聚类”的异常挖掘方法。该算法首先应用 TwoStep 聚类算法对数据集进行聚类分析（TwoStep 算法在第 4 章中有详细介绍），将数据集划分为若干个聚类（或称“对等组”），然后对每一个样本，计算与其最近的聚类的距离，并根据距离的大小计算出一个“异常指数”，来说明这个样本到底有多么“异常”。最后，用户可以通过设定异常指数的阈值，将那些大于阈值的样本选择出来作为异常数据。

执行异常检测算法，可以完成两个任务：第一，从数据集中确定哪些样本是异常样本；第二，对每一个异常样本，分析是哪些属性的取值导致了该样本成为了一个异常样本。

该算法对数据的格式要求很低，数据集可以同时包含连续属性和离散属性，也可以有缺失值。不过算法假定连续属性的取值服从正态分布，离散属性的取值分布服从多项分布。

整个算法分为 3 个步骤：

建模（modeling）：用聚类算法对样本进行划分。

打分（scoring）：对每一个样本，计算它与其所在聚类的距离，从而计算出它的异常指数。然后将所有样本根据异常指数的大小降序排列。最后，异常指数最大的一部分样本被确定为异常样本。

推理（reasoning）：对每一个异常样本，计算它的每一个属性的“变量偏离指数”（Variable Deviation Index，VDI），用来衡量该属性到底做出了多大的贡献使得该样本成为了一个异常样本。然后对每个异常样本，将所有属性按照 VDI 的大小降序排序，VDI 最大的那几个属性及其取值被认为是该样本成为异常样本的重要原因。

在算法的描述中，用到了下列表示符号：

ID：数据集中的样本号。

n：数据集中的样本数量。

X_{ok}，$k=1,2,\cdots,K$：全部输入属性组成的集合。

X_{K+1}：在数据分析过程中产生的一个新的属性变量，用来表示每个样本中有缺失值

的属性数量占全部属性数量的比例。

X_k，k=1,2,…,K ：在处理了缺失值之后数据集中的每个属性。

M_k，k=1,2,…,K ：对于一个连续属性 X_k，M_k 表示该属性所有样本值的平均值，即总体均值。

SD_k，k=1,2,…,K ：对于一个连续属性 X_k，SD_k 表示该属性的总体标准差。

H，或者 H 的边界[H_{min}, H_{max}]：H 是预定义的聚类数量。或者，最少 H_{min} 个聚类，最多 H_{max} 个聚类。

n_h，h=1,2,…,H ：聚类 h（h=1,2,…,H）中的样本数量。

p_h，h=1,2,…,H ：聚类 h（h=1,2,…,H）中的样本量和总体样本量的比例，$p_h=n_h/n$。

M_{hk}, k = 1, …, K+1, h = 1,…, H ：对于一个连续属性 X_k，M_{hk} 表示在聚类 h 中所有样本的 X_k 的平均值。如果 X_k 是一个离散属性，M_{hk} 表示在聚类 h 中，样本数量最多的 X_k 的值（众数）。

SD_{hk}, $k \in \{1, \cdots, K+1\}$, h= 1, …, H ：对于一个连续属性 X_k，SD_{hk} 表示聚类 h 中 X_k 的标准差。

$\{n_{hkj}\}$, $k \in \{1, \cdots, K\}$, h =1, …, H, j = 1, …, J_k ：当 X_k 是一个离散属性，有 J_k 个不同的取值（表示不同的类别），n_{hkj} 表示在聚类 h 中属于 j 类（X_k=j）的样本数量。

m：一个调整的权重，用来平衡连续属性和离散属性对建模的影响，默认值为 6。

VDI_k, k = 1, …, K+1：变量偏差指数（Variable Deviation Index），某个样本的某个属性 X_k 与该样本所在聚类的该属性的范数之间的偏差指数。

GDI：一个样本的组偏差指数（Group Deviation Index），是一个对数-似然距离 $d(h,s)$，是所有属性的变量偏差指数{ VDI_k, k = 1, …, K+1}的和。

异常指数（Anomaly Index）：一个样本的异常指数，是指该样本的 GDI 除以该样本所在聚类的平均 GDI。

变量贡献率（Variable Contribution Measure，VCM_k）：对于某个样本的某个变量 X_k，用它的 VDI_k 除以它的 GDI。

$\mathrm{pct}_{anomaly}$ or $n_{anomaly}$ ：$\mathrm{pct}_{anomaly}$ 是预先定义的异常样本的比例。或者，$n_{anomaly}$ 是预先定义的异常样本的数量（一个正整数）。

$\mathrm{Cutpoint}_{anomaly}$：一个预定义的分割点，用于将全部样本按照分割点来划分为异常样本和非异常样本。

$k_{anomaly}$：一个预定义的整数阈值 $1 \leqslant k_{anomaly} \leqslant K+1$，对于一个样本的全部属性，要分析其中 $k_{anomaly}$ 个影响力最大的属性。

1. 建模阶段

在建模阶段要执行下列任务：

（1）数据集的格式化

有的样本的连续属性的值太大（大于 10^{150}），那么删除这个样本。

有的样本的每个属性的值都缺失，那么删除这个样本。

有的属性，所有样本的该属性的取值都相同（或者缺失），那么删除这个属性。

（2）缺失值的处理（可选步骤）

对于数据集中的每个属性 X_{ok}，k=1,2,⋯,K，如果 X_{ok} 是一个连续属性，用该属性所有的有效的值来计算该属性的总体平均值 M_k 和总体标准差 SD_k。用计算出来的总体平均值来代替所有的缺失值。如果 X_{ok} 是一个离散属性，用一个新的值 missing value 来代替所有的缺失值，它被作为了一个有效的属性值。数据集$\{X_{ok}\}$被以上处理完之后，用$\{X_k\}$来表示。

（3）创建"缺失值比例"变量（可选步骤）

创建一个新的属性变量 X_{K+1}，用来表示每个样本中有缺失值的属性数量占全部属性数量的比值。

（4）聚类划分

用经过上面处理后得到的属性变量$\{X_k, k = 1, \cdots, K+1\}$作为输入变量，用 TwoStep 聚类算法来构建一个聚类模型，从而将样本分配到某个聚类中。

（5）生成并存储各种统计量

对每一个连续属性 X_k，计算并存储它的总体平均值 M_k 和总体标准差 SD_k。

对每一个聚类 h（h=1,2,⋯, H），计算它的样本量 n_h。如果 X_k 是一个连续变量，计算聚类 h 中 X_k 的平均值 M_{hk} 和标准差 SD_{hk}；如果 X_k 是一个离散属性，有 j 个不同的取值，计算每个取值在聚类 h 中的频度 n_{hkj}，并保存该属性在聚类 h 中的众数 M_{hk}。这些统计量将被用于计算一个聚类 h 和一个给定的样本 s 之间的对数-似然距离 $d(h,s)$。

2．打分阶段

整个算法的核心部分在这一阶段。本阶段给测试数据集（或者训练数据集）的每个样本打分，也就是计算每个样本的异常指数以及各属性的变量贡献率。

这个阶段执行下列任务：

（1）对测试数据的格式进行检查

测试数据中必须包含有训练数据中的所有输入字段$\{X_{ok}, k = 1, 2, \cdots, K\}$，而且这些字段的格式必须也与训练数据的格式相同。

在测试数据中，如果某个样本的某个离散属性的值在训练数据中没有出现过，那么这个样本要被删除。例如，在训练数据中，region 这个离散属性的所有取值为 *IL*，*MA* 和 *CA*，但是在测试数据中有一个样本其 region 的取值为 *FL*，那么这个样本要被排除。

（2）缺失值的处理

对每一个输入变量 X_{ok}，k=1,2,⋯,K，如果 X_{ok} 是一个连续属性，用该属性所有的有效的值来计算该属性的总体平均值 M_k 和总体标准差 SD_k。用计算出来的总体平均值来代替所有的缺失值。如果 X_{ok} 是一个离散属性，用一个新的值 missing value 来代替所有的缺失值，它被作为了一个有效的属性值。

（3）创建"缺失值比例"变量（视建模阶段是否有，可选）

如果在建模阶段创建了 X_{K+1}，那么在打分阶段也对测试数据生成 X_{K+1}。

（4）对测试数据进行聚类

用生成的聚类模型，对测试数据中的每个样本进行聚类，把每个样本分配到一个聚

类当中。

（5）计算变量偏差指数 VDI_k 和组偏差指数 GDI

给定一个样本 s，以及与其最近的聚类 h，样本 s 的组偏差指数 GDI 其实就是 s 与 h 的对数似然距离 $d(h,s)$，用来衡量该样本与聚类 h 的相似程度，距离越大，表示相似度越小，反之则相似度越大。而 $d(h,s)$是由样本的每一个属性值 X_k 与聚类中相应的属性值之间的距离分量 $d_k(h,s)$累加而成的。这里 $d_k(h,s)$就称为变量 X_k 的变量偏差指数 VDI_k。因此，要计算样本 s 的组偏差指数 GDI，就必须先计算 s 的所有变量偏差指数$\{VDI_k，k = 1, \cdots, K+1\}$。

在 TwoStep 算法中阐述了计算两个聚类之间对数似然距离的方法，这里计算一个样本与一个聚类之间的对数似然距离时，可以将样本当做只有一个样本的聚类来处理。

那么，属性 X_k 相对于聚类 h 的变量偏差指数 VDI_k 的计算方法为：

① 当属性变量 X_k 是一个连续属性时

$$d_k(h,s)=\frac{1}{2}[-N_h\log(\Delta_k+\hat{\sigma}_{hk}^2)-N_s\log(\Delta_k+\hat{\sigma}_{sk}^2)+N_{<h,s>}\log(\Delta_k+\hat{\sigma}_{<h,s>k}^2)]$$

其中：

N_h 是聚类 h 中（不包含样本 s）的样本数量；

$\Delta_k=\frac{\hat{\sigma}_k^2}{m}$ 是一个正的调节因子，用来避免当聚类只有一个样本时对数值无意义的情况（这里的 m 由用户自行定义，默认值为 6，$\hat{\sigma}_k^2$ 是整个数据集的 X_k 的样本方差）；

$\hat{\sigma}_{hk}^2$ 是聚类 h 中的 X_k 的样本方差；

N_s 是聚类 s 中的样本数量，显然，N_s=1；

$\hat{\sigma}_{sk}^2$ 是聚类 s 中的 X_k 的样本方差，显然，$\hat{\sigma}_{sk}^2$ =0；

$N_{<h,s>}$ 是聚类 h 包含了样本 s 之后的样本数量，显然，$N_{<h,s>}=N_h+1$；

$\hat{\sigma}_{<h,s>k}^2$ 是聚类 h 包含了样本 s 之后 X_k 的样本方差。

因此，上式可化简为：

$$d_k(h,s)=\frac{1}{2}[-N_h\log(\Delta_k+\hat{\sigma}_{hk}^2)-\log(\Delta_k)+(N_h+1)\log(\Delta_k+\hat{\sigma}_{<h,s>k}^2)]$$

② 当属性变量 X_k 是一个离散属性时，

$$d_k(h,s)=-N_h\hat{E}_{hk}-N_s\hat{E}_{sk}+N_{<h,s>}\hat{E}_{<h,s>k}$$

其中：

N_h 是聚类 h 中（不包含样本 s）的样本数量；

$\hat{E}_{hk}=-\sum_{l=1}^{L_k}[\frac{N_{hkl}}{N_h}\log(\frac{N_{hkl}}{N_h})]$ 是聚类 h 中 X_k 的所有取值的期望频次对数统计量，L_k 表示 X_k 有 L_k 个不同的取值，N_{hkl} 表示聚类 h 中 X_k 等于第 l 个值的样本数量；

N_s 是聚类 s 中的样本数量，显然，N_s=1；

$\hat{E}_{sk}$ 聚类 s 中 X_k 的期望频次对数统计量，由于 s 中只有一个样本，所以 $\hat{E}_{sk}$ =0；

$N_{<h,s>}$ 是聚类 h 包含了样本 s 之后的样本数量，显然，$N_{<h,s>}=N_h+1$；

$\hat{E}_{<h,s>k} = -\sum_{l=1}^{L_k}\left[\frac{N_{<h,s>kl}}{N_h+1}\log\left(\frac{N_{<h,s>kl}}{N_h+1}\right)\right]$是聚类 h 包含了样本 s 之后 X_k 的期望频次对数统计量，$N_{<h,s>kl}$ 表示聚类 h 包含了样本 s 之后 X_k 等于第 l 个值的样本数量。

因此，上式可化简为

$$d_k(h,s) = -N_h\hat{E}_{hk} + (N_h+1)\hat{E}_{<h,s>k}$$

在计算了所有属性变量的偏差指数$\{VDI_k, k=1, 2,\cdots, K+1\}$之后，即可计算出样本的组偏差指数 GDI

$\text{GDI} = d(h,s) = \sum_{k=1}^{K^A+K^B} d_k(h,s)$，其中，$K^A$ 是样本的连续属性的个数，K^B 是样本的离散属性的个数。

（6）计算异常指数和变量贡献

一个样本 s 的异常指数，用来衡量该样本与其所在聚类 h（包含样本 s）中的其他样本相比，到底有多么得异常。它是样本 s 的 GDI 除以聚类 h 中所有样本的平均 GDI 所得的比值

$$\text{AnomalyIndex} = \frac{\text{GDI}_s}{\text{mean}(\text{GDI}_h)}$$

异常指数越大，表示样本越异常。通常情况下，异常指数值小于 1 甚至小于 1.5 的观测值都不会被视为异常值，因为该偏差与平均值相同或者只是大一点。但是，指数值大于 2 的观测值有可能是异常观测值，因为该偏差至少是平均值的两倍。

一个属性变量 X_k 的变量贡献率，是对于单个的样本 s 而言的。它是样本 s 中属性变量 X_k 的 VDI_k 除以样本的 GDI_s 所得的比值

$$\text{VCM}_k = \frac{\text{VDI}_k}{\text{GDI}_s}$$

变量贡献率用来衡量一个属性对于一个样本异常程度的影响程度。一个属性的变量贡献率越大，表示该属性使得样本趋于异常的影响力也越大。

对于一个异常指数比较大的样本，可以通过变量贡献率，来分析到底是哪些属性的取值使得该样本成为了一个异常样本。

3．推理阶段

通过上面的步骤，已经计算出了一些相关的指数，包括组偏差指数、异常指数、变量偏差指数以及变量贡献率。在推理阶段，要确定出哪些样本是异常样本，并给出将它们确定为异常样本的理由。

（1）确定异常样本

将全部样本按照异常指数的大小降序排列。然后根据异常分割点 $\text{Cutpoint}_{\text{anomaly}}$ 来将样本划分为异常或者非异常的样本。可以有 3 种确定分割点的方法：

① 最小异常指数。大于最小异常指数的样本，被确定为异常样本。

② $\text{pct}_{\text{anomaly}}$：按照异常指数降序排列后，最前面的 $\text{pct}_{\text{anomaly}}$ %的样本被确定为异常样本。

③ $n_{anomaly}$：按照异常指数降序排列后，最前面的 $n_{anomaly}$ 个样本被确定为异常样本。这里 $pct_{anomaly}$ 和 $n_{anomaly}$ 的值由用户预先设定。

（2）异常的原因分析

对每一个异常样本，将它的所有属性按照 VDI_k 的值降序排列。前 $k_{anomaly}$ 个属性的属性名、属性值等被列出，作为其异常的原因。这里，$k_{anomaly}$ 的值由用户预先设定。

6.2.3　在 Clementine 中应用异常检测

本小节演示在 Clementine 中应用异常检测节点来从数据集中筛选出异常样本的例子。本例参考了 Clementine 自带的一个应用程序示例，用于识别农业发展财政补贴申请中的欺诈行为。

训练数据集存放在文件 grantfraudN.db 中，该文件保存在 Clementine 安装目录下的 Demos 文件夹中。这是一个记录了有关农业发展财政补贴申请信息的数据集，包括 10 个字段（样本 ID、申请人姓名、地区、田地大小、降雨量、田地质量、田地收入、主要农作物、申请类型、申请金额），共 300 个样本。现在，需要从中筛选出 10 个最异常的样本，为后续的异常数据分析做数据准备。

完整的数据流如图 6.13 所示。

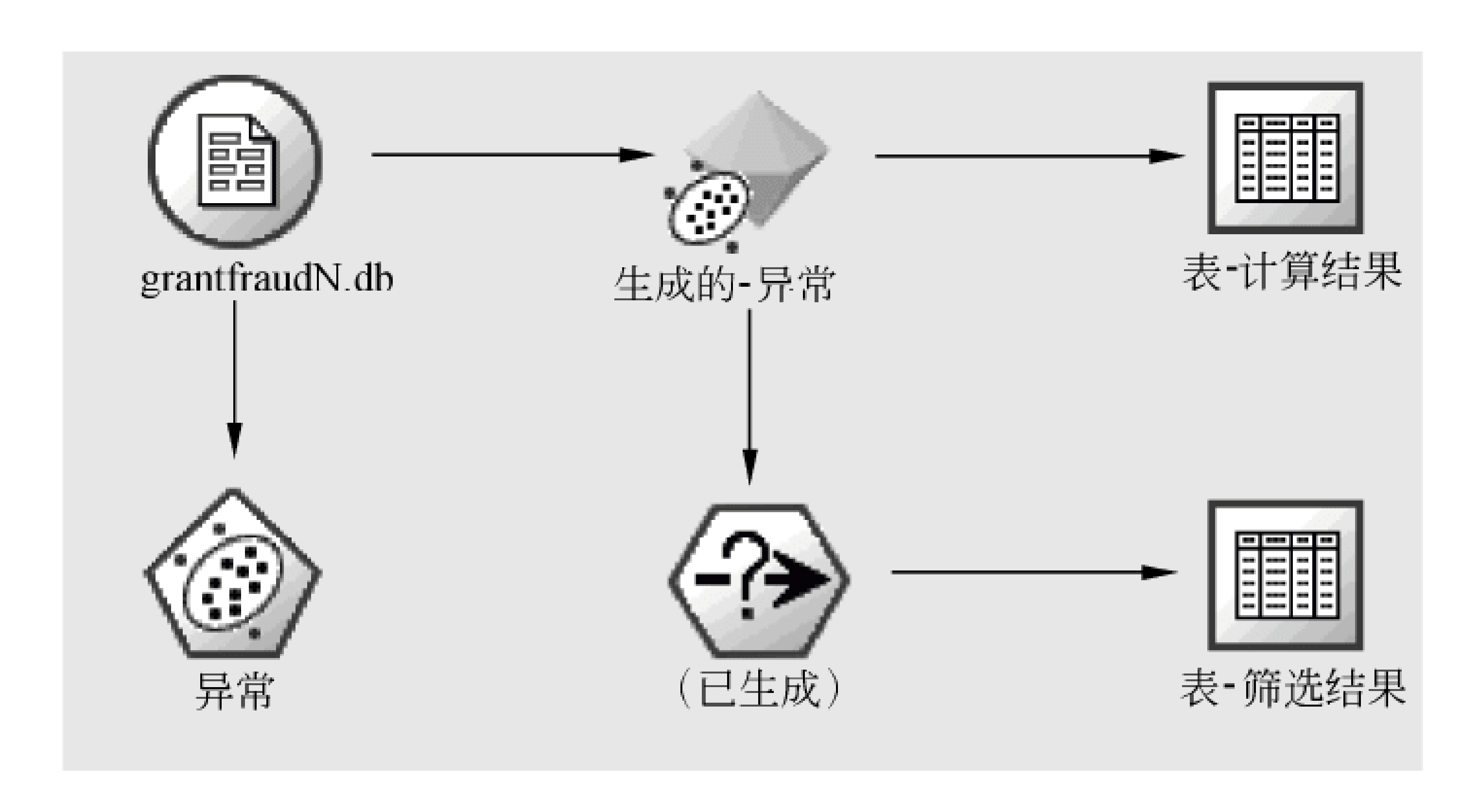


图 6.13　异常数据挖掘数据流

1. 生成模型

首先，将“数据源”中的“可变文件”节点添加到数据流区域，并将 grantfraudN.db 文件加载到该节点，然后对该节点进行编辑，如图 6.14 所示，在“类型”标签下，单击“读取值”按钮读取数据，然后将字段 ID 和 name 的方向设置为“无”，类型为“无类型”，即这两个字段不参与建模，因为它们与样本是否异常没有任何关系。所有其他字段的方向应设置为“输入”，这样它们将作为输入内容包含在异常检测模型中。

向数据流中添加建模节点“异常”节点，建立从数据源 grantfraudN.db 节点到“异常”节点的连接，然后对“异常”节点进行设置，在节点编辑窗口的“模型”标签下，设置如图 6.15 所示。



图 6.14　编辑数据源节点

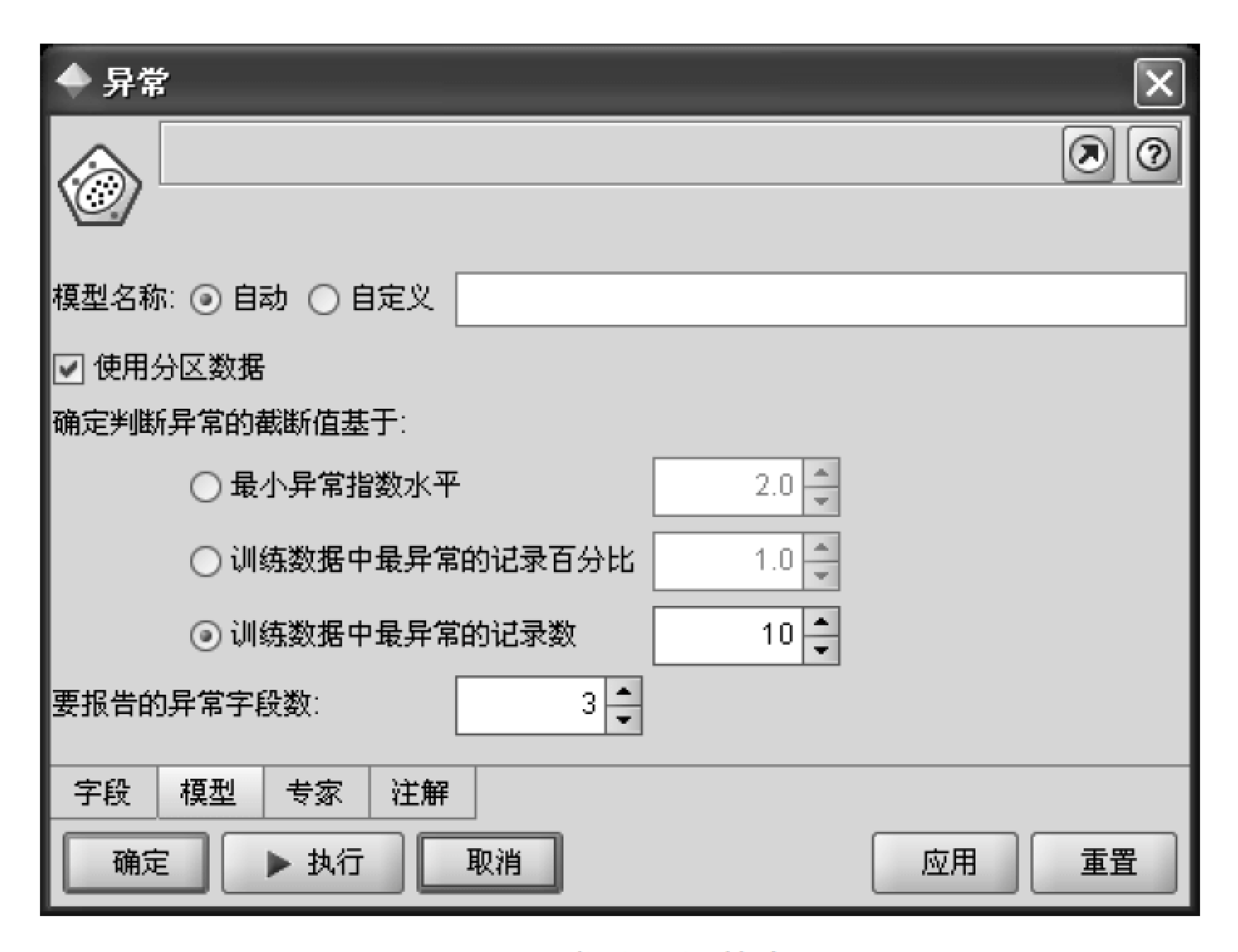


图 6.15　异常挖掘的基本设置

确定判断异常的截断值基于（Determine Cutoff Value For Anomaly Based On）：也就是设置异常分割点，如前所述，有 3 种确定分割点的方法。

（1）最小异常指数水平（Minimum Anomaly Index Level）：即最小异常指数阈值，默认值为 2，异常指数大于等于 2 的样本为异常样本。

（2）训练数据中最异常记录的百分比（Percentage of Most Anomalous Records In The Training Data）：设置 $pct_{anomaly}$ 的值，默认为 1，即异常指数最高的 1%个样本为异常样本。

（3）训练数据中最异常的记录数（Number of Most Anomalous Records In The

Training Data）：设置 $n_{anomaly}$ 的值，默认为 10，即异常指数最高的 10 个样本为异常样本。

这里选择第 3 项，即筛选出 10 个异常样本。

要报告的异常字段数（Number of Anomaly Fields To Report）：设置 $k_{anomaly}$ 的值，默认值为 3，即对每一个样本列出 3 个变量偏差指数最大的属性。这里保持默认设置。

切换到编辑窗口的“专家”标签下，设置如图 6.16 所示。

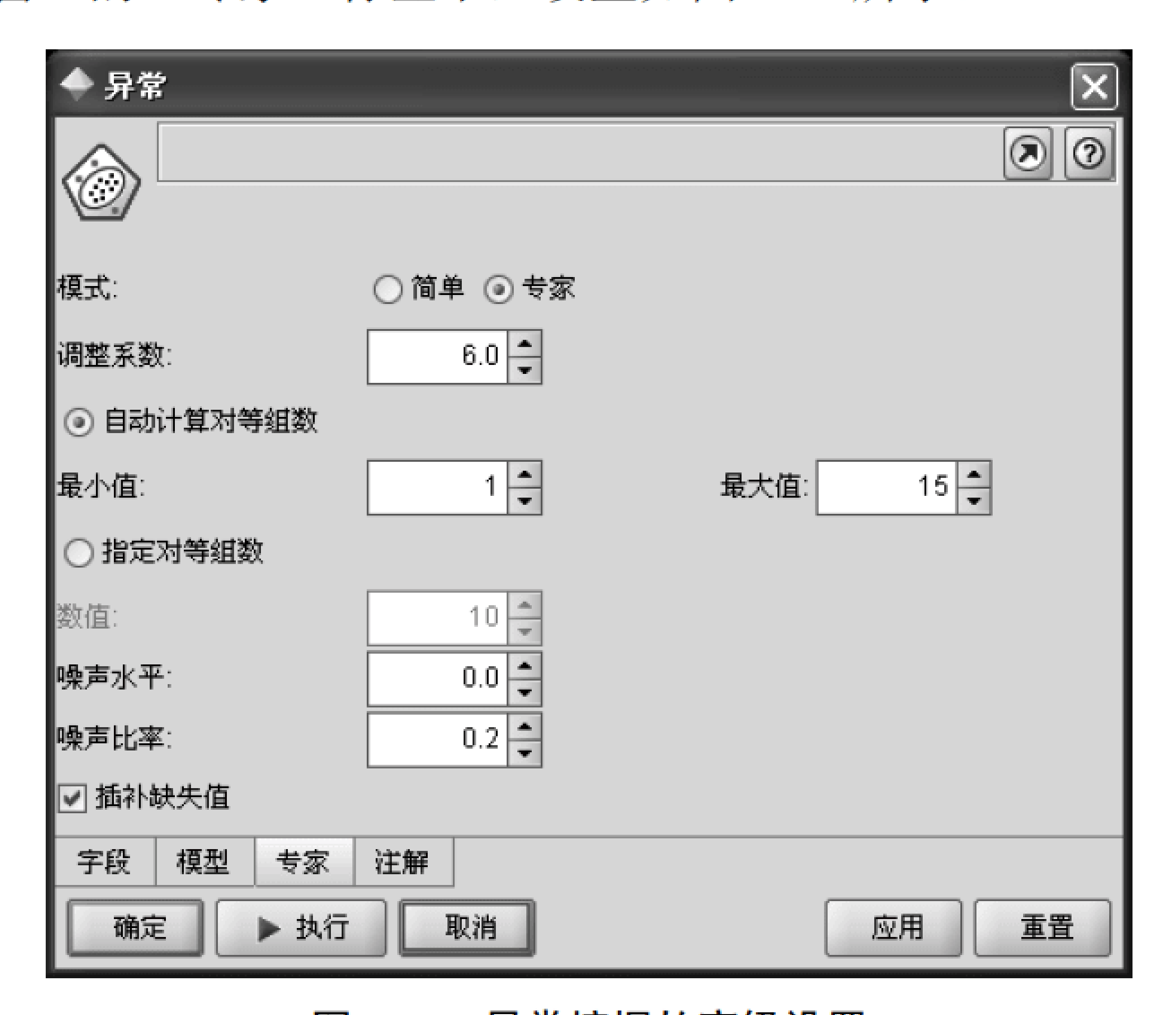


图 6.16　异常挖掘的高级设置

选择“专家”模式，然后对下列参数进行设置：

调整系数（Adjustment Coefficient）：在计算连续属性 X_k 的变量偏差指数 VDI_k 时，采用了一个正的调节因子 $\Delta_k = \frac{\hat{\sigma}_k^2}{m}$，这里的调整系数就是参数 m，默认值为 6。m 用于平衡在计算距离时赋予连续属性和离散属性的相对权重的数值。值越大，连续属性的影响也越大。它必须为非 0 值。本例中保持默认设置。

自动计算对等组数（Automatically Calculate Number Of Peer Groups）：异常检测可以通过在聚类过程中生成不同数量的聚类而形成多种可能的解决方案，并选择训练数据的最佳的聚类数量。用户可以设置聚类数的“最小值”和“最大值”（“最小值”和“最大值”就是前文中提到的 H_{min} 和 H_{max}）使系统在一个范围内搜索可能的解决方案。当然，范围越大，消耗的处理时间也随之增加。

指定对等组数（Specify Number Of Peer Groups）：如果知道模型中应包含聚类的数量，可选中此选项并输入相应的数值。由于无须搜索可能的解决方案，选中此选项可提高性能。

噪声水平（Noise Level）和噪声比率（Noise ratio）：设置这两个参数是为了确定在 TwoStep 聚类的过程中如何处理离群值。

噪声水平：指定值必须处于 0～0.5 之间。此设置只有在以下情况中才有用：CF 树在生长期间被充满，即该树的叶节点无法接收更多的观测值且叶节点无法分裂。如果 CF 树被充满且信噪等级设置为 0，则阈值 T 将增大且 CF 树将用所有观测值重新生长（详见

第 4.3 节）。创建最终聚类后，先前那些无法被分配到聚类的值将被标记为离群值。离群值聚类将被赋予一个识别号“–1”。离群值聚类不包含在聚类数的计数中，也就是说，如果指定 *n* 个聚类和噪声处理，则算法将输出 *n* 个聚类和一个噪声聚类。在实际应用中，增大此值可使算法更有可能将异常记录分配到普通聚类中，而不是将它们分配到独立的离群值聚类中。

如果 CF 树被充满且信噪等级大于 0，则 CF 树把稀疏叶子节点中的所有数据放入其自身的噪声叶子节点后再重新生长（如果叶子节点中的观测值数量与最大叶子节点中的观测值数量的比率小于噪声水平，则该叶子节点将被认定为稀疏叶子节点）。树创建完成后，可能的话，离群值将被放置在 CF 树中。如果未放在树中，在第二步聚类中将丢弃这些离群值。

噪声比率：在噪声缓冲区中内存的分配比例，此值必须处于 0～0.5 之间。如果将特定观测值插入树的叶子节点后，所产生的紧密度小于阈值，叶子节点将不再分裂。如果紧密度超过阈值，叶子节点将分裂，也就是产生一个新的小聚类。实际上，提高此设置值将可能导致算法容易更快速地创建较简单的树。

插补缺失值（Impute Missing Values）：该选项实际上是指出缺失值的处理方式。如果选中该选项，对于连续型属性，用均值来填充缺失值。对于离散型属性，用一个新的值 missing value 来填充所有的缺失值，它被作为了一个有效的属性值。如果取消选中此选项，则任何带有缺失值的记录都将从分析中剔除。

以上设置结束后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示生成的“异常”模型。

2．浏览和应用模型

将生成的“异常”模型节点从管理器窗口拖入到数据流区域中，双击该节点，即可打开编辑窗口，对模型进行浏览，如图 6.17 所示。

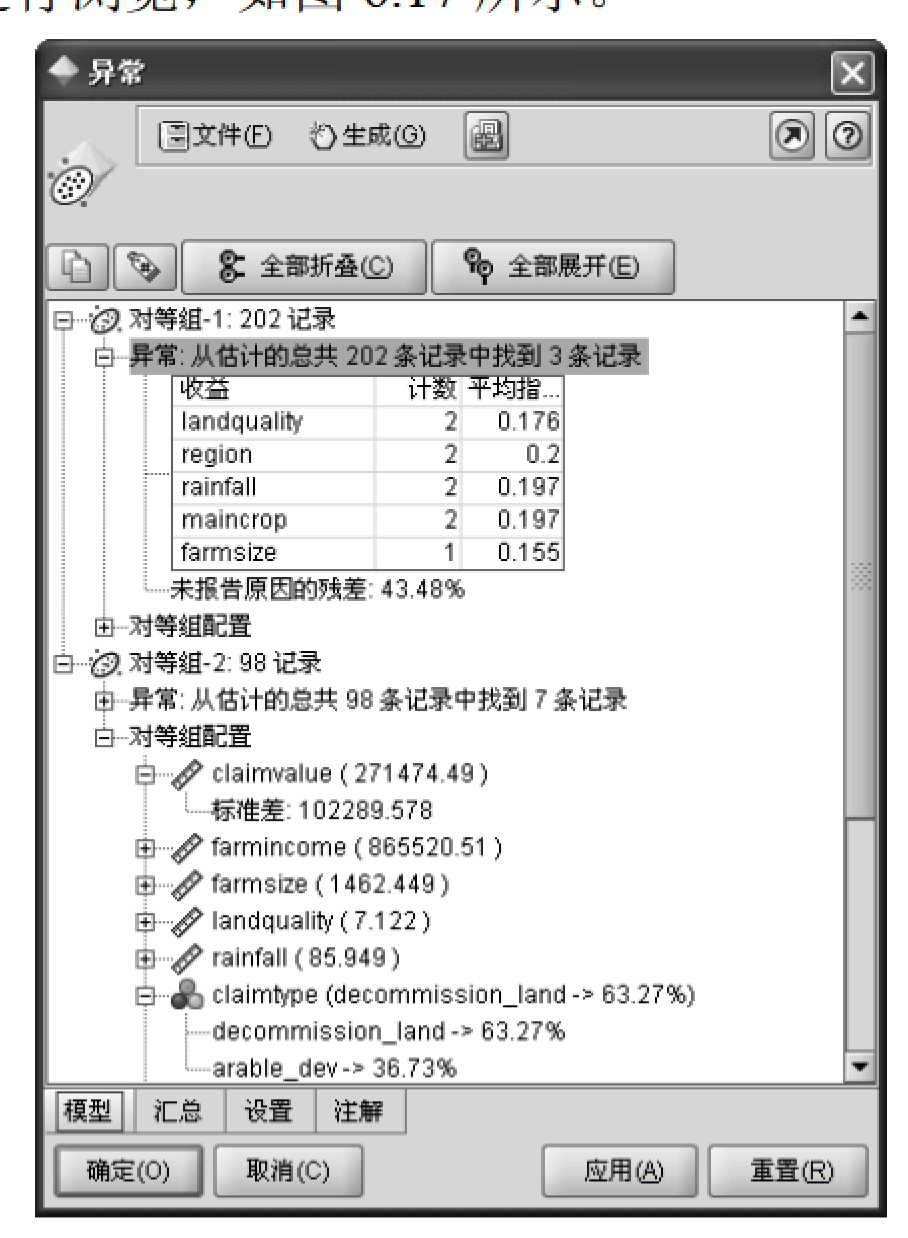


图 6.17　模型的详细信息

在“模型”标签下，可以看到，所有记录被划分为了两个对等组（聚类），对等组-1中包含有 202 个记录，其中有 3 条记录被确定为异常。对等组-2 中包含有 98 个记录，其中有 7 条记录被确定为异常。

这里，显示了每个对等组中异常记录的统计信息表，其中包括“收益”（contribution）、“计数”（count）和“平均指数”（Average Index），罗列出了变量偏差指数较高的一些属性及相关统计信息。

例如，对于对等组-1 中的 3 个异常记录，在筛选结果（见后续内容）中可以找到这 3 个异常记录以及每个记录的 3 个变量偏差指数最大的属性：

异常记录 1：rainfall（0.230）；region（0.219）；landquality（0.193）

异常记录 2：maincrop（0.236）；rainfall（0.163）；farmsize（0.155）

异常记录 3：region（0.181）；landquality（0.160）；maincrop（0.158）

在“收益”一列中，列出了上面的这些属性；其中 landquality 出现了两次，所有它的“计数”为“2”，而它的“平均指数”就等于这两次的变量偏差指数的平均值（0.193+0.160）/2=0.176。同理，计算并显示了其他属性的“计数”和“平均指数”。

另外，在“对等组配置”下，还显示了在该对等组中，所有属性的统计信息：

对于连续属性，显示了属性在该组中的均值和标准差。例如对等组-2 中，属性 claimvalue 的均值为 271474.49，标准差为 102289.578。

对于离散属性，显示了属性在该组中的每个取值的样本比例。例如属性 claimtype，有 63.27%的记录该属性取值为 decommission_land，有 36.73%的记录该属性取值为 arable_dev。

在窗口的“汇总”标签下，显示了关于本次建模的一些汇总信息，如图 6.18 所示，其中显示了异常指数的截断值，本例的异常指数截断值为 1.35034，即异常指数大于 1.35034 的记录被确定为了异常记录。

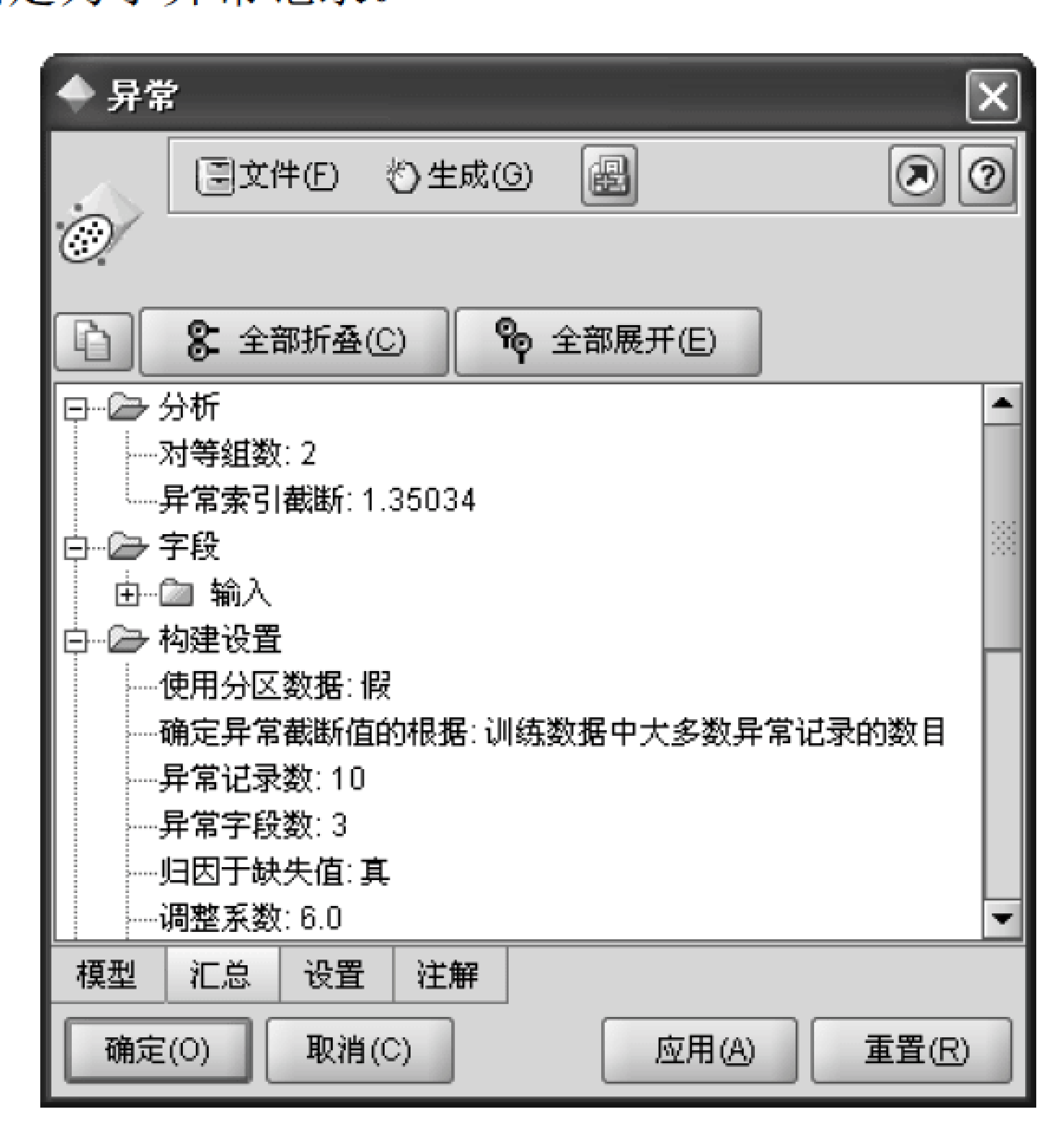


图 6.18 模型的汇总信息

在窗口的“设置”标签下，要对模型进行一些设置（如图 6.19 所示），这是为了在后面当用该模型对测试数据进行异常检测时对检测的输出结果进行控制。



图 6.19　设置模型以控制输出结果

指出异常记录（Indicate Anomalous Records With）：这个设置指出在对测试数据进行异常检测后，在输出结果中对异常记录的处理方式，有 3 个选项：

（1）**标志和指数（Flag And Index）**：在输出结果中，创建一个标志字段，任何异常指数超过截断值的记录，其标志字段的值将被设置为真（T），否则设置为假（F）。同时，列出每个记录的异常指数。

（2）**仅限于标志（Flag Only）**：创建标志字段，但不报告每条记录的异常指数。

（3）**仅限于指数（Index Only）**：报告异常指数但不创建标志字段。

要报告的异常字段数（Number of Anomaly Fields To Report）：设置数值 $k_{anomaly}$，将每个记录中异常指数最大的 $k_{anomaly}$ 个属性及其异常指数列出来。默认值为 3。

丢弃记录（Discard Records）：如果需要在后续的数据分析中仅仅关注异常数据或者非异常数据，可以选择此选项，然后选择“非异常”，可将所有非异常记录丢弃（或者“异常”，将所有异常记录丢弃）。这里不选择“丢弃记录”，即输出所有记录。

在“注解”标签下，选择“自定义”，并在其后的文本框中输入“生成的-异常”，以把该节点的名称更改为“生成的-异常”。

最后，单击“确定”按钮，结束设置。现在可以用“生成的-异常”节点来对测试数据进行异常检测了，这里就直接对训练数据进行测试。

建立从 grantfraudN.db 节点到“生成的-异常”节点的连接，然后在“生成的-异常”节点后面添加一个“表”节点，并建立从“生成的-异常”节点到该“表”节点的连接，双击“表”节点，打开编辑窗口，在窗口的“注解”标签下，将名称自定义为“表-计算结果”，然后执行“表”节点，结果如图 6.20 所示。

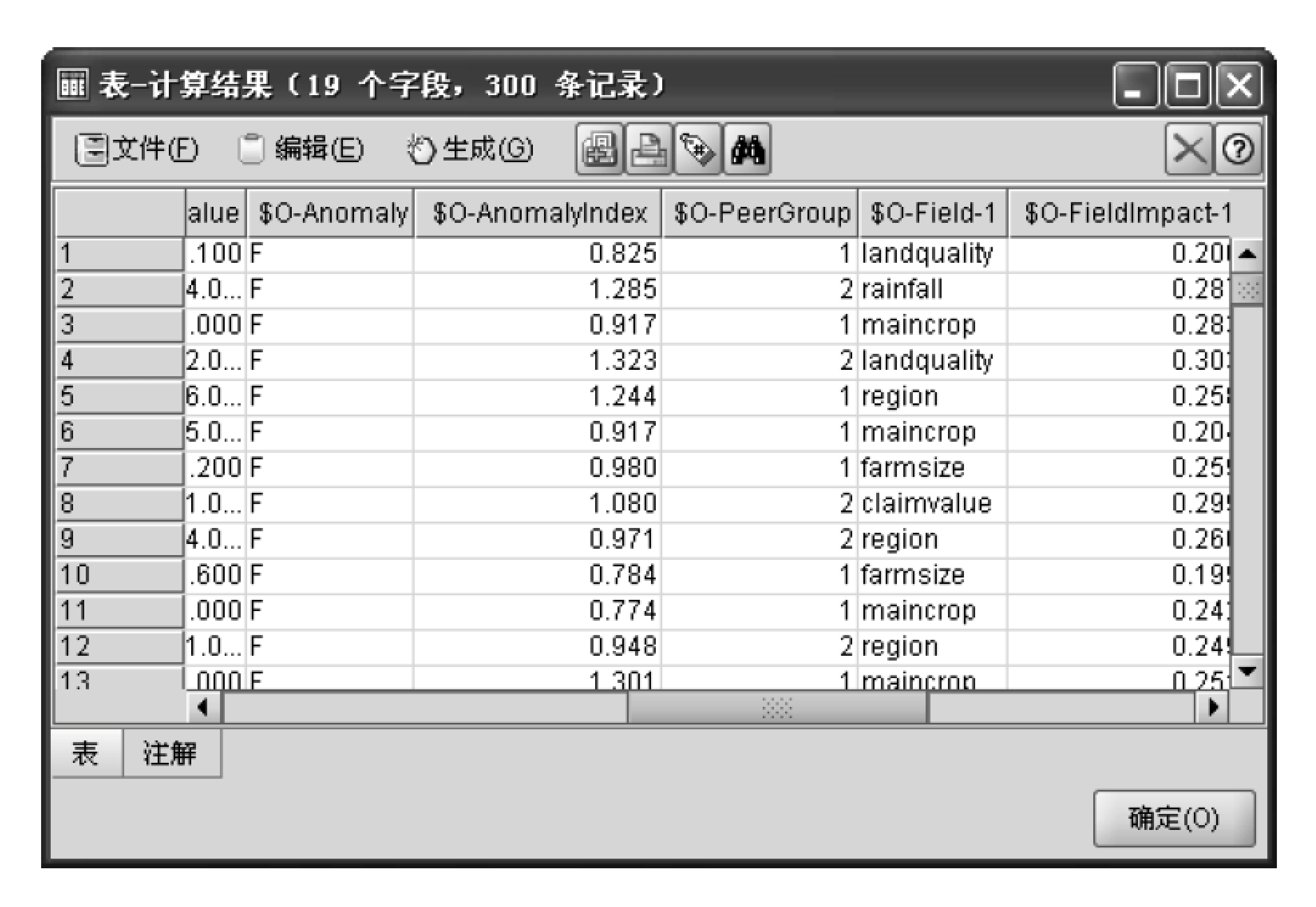


图 6.20　异常检测结果

可以看到，在原有的 10 个字段基础上，输出结果中又多出来若干个字段：

$O-Anomaly：标志字段，若为异常记录，则显示“T”，否则为“F”。

$O-AnomalyIndex：记录的异常指数值。

$O-PeerGroup：记录被划分到的聚类号。

$O-Field-n：记录中变量偏差指数最大的第 *n* 个属性的名称。

$O-FieldImpact-n：$O-Field-n 所指属性的变量偏差指数。

下面，根据模型生成选择节点，把所有异常记录筛选出来。

双击“生成的-异常”节点，在“生成”菜单下单击“选择节点”按钮，即可在数据流区域生成一个“:生成的”选择节点，双击该节点，如图 6.21 所示。

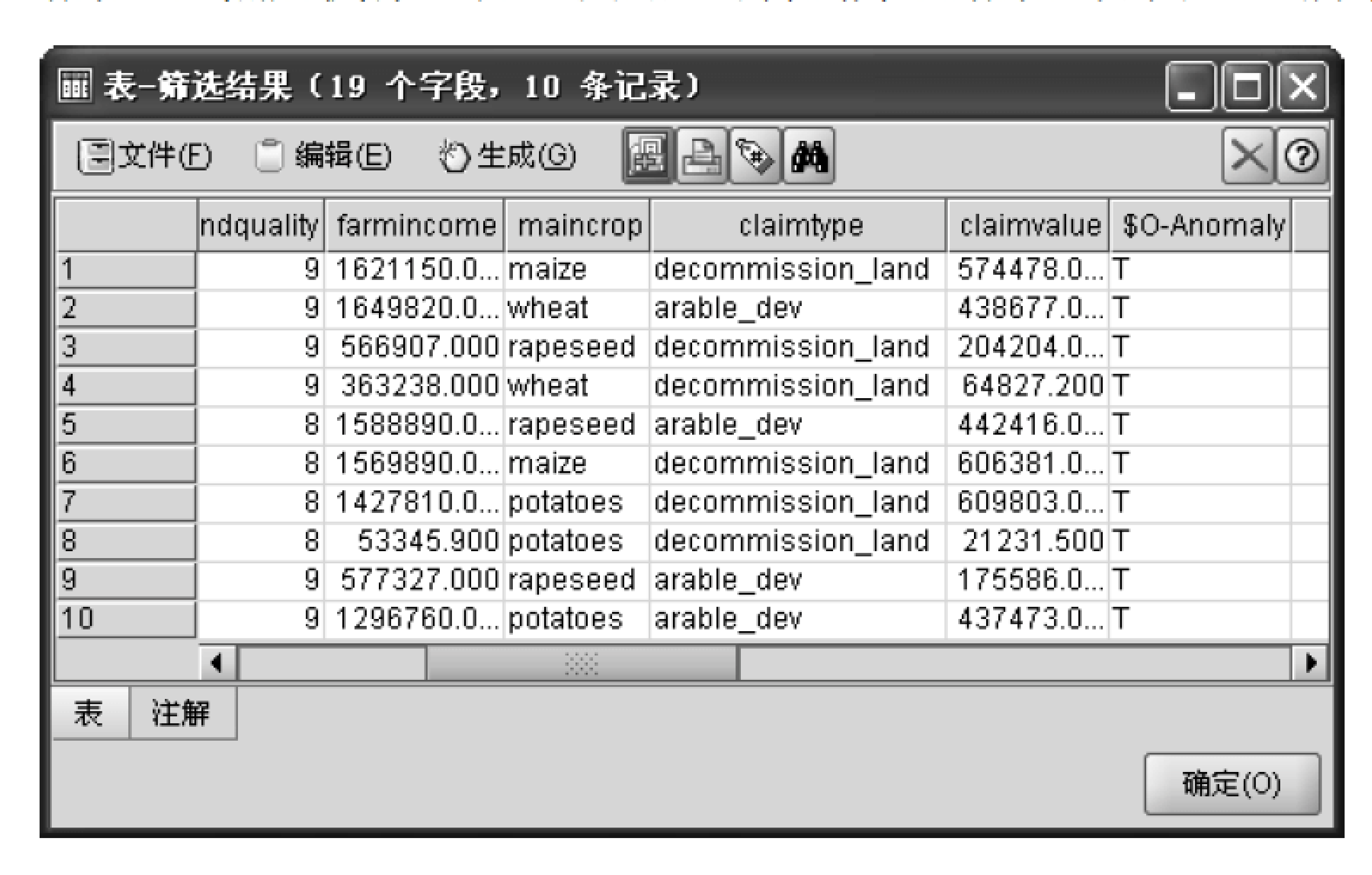


图 6.21　生成的选择节点

可以看到在条件栏中有“'$O-AnomalyIndex'>=1.35034”，即把异常指数大于等于 1.35034 的记录筛选出来。

建立从节点“生成的-异常”到节点“:生成的”连接，并在“:生成的”后面添加“表”节点，建立到“表”节点的连接，双击“表”节点后，在编辑窗口中将“表”节点更名

为“表-筛选结果”，然后执行“表”节点，可得到筛选结果，如图 6.22 所示。



	ndquality	farmincome	maincrop	claimtype	claimvalue	$O-Anomaly
1	9	1621150.0...	maize	decommission_land	574478.0...	T
2	9	1649820.0...	wheat	arable_dev	438677.0...	T
3	9	566907.000	rapeseed	decommission_land	204204.0...	T
4	9	363238.000	wheat	decommission_land	64827.200	T
5	8	1588890.0...	rapeseed	arable_dev	442416.0...	T
6	8	1569890.0...	maize	decommission_land	606381.0...	T
7	8	1427810.0...	potatoes	decommission_land	609803.0...	T
8	8	53345.900	potatoes	decommission_land	21231.500	T
9	9	577327.000	rapeseed	arable_dev	175586.0...	T
10	9	1296760.0...	potatoes	arable_dev	437473.0...	T

图 6.22　输出所有异常记录

若想对异常记录做进一步的分析，可以在“:生成的”节点之后添加其他建模节点来继续分析。

第 7 章 统 计 模 型

7.1 线性回归

7.1.1 线性回归的基本原理

1. 基本概念

客观事物相互依存，且具有存在和发展的内在关联。这种关联反映在数据上，就表现为变量和变量之间的关系。通常，变量之间的关系可以划分为两类：

① 完全确定的函数关系。如果一个变量 y 的取值可以通过另一个变量 x 或者一组变量（$x_1,x_2,\cdots,x_p$）以某种形式唯一地确定，则它们之间就是函数关系，表示为 $y=f(x)$或者 $y=f(x_1,x_2,\cdots,x_p)$。例如圆面积与圆半径的关系，就是函数关系。

② 非确定的相关关系。如果一个变量 y 的取值受到另一个变量 x 或者一组变量（$x_1,x_2,\cdots,x_p$）的影响，但给定 x 或者（$x_1,x_2,\cdots,x_p$）的值时，y 的取值并不是唯一确定的，那么变量 y 与 x 或（$x_1,x_2,\cdots,x_p$）之间为相关关系，表示为 $y=f(x,\varepsilon)$或者 $y=f(x_1,x_2,\cdots,x_p,\varepsilon)$。例如收入与支出之间的关系，两个收入相同的人，他们的支出往往是不同的，但即便如此，从宏观上来看，收入和支出之间又是有一定关联的，这种关系就是相关关系。

自然现象和社会现象中的变量关系，大量地表现为相关关系。线性回归分析正是一种用于发现变量之间相关关系的统计学方法，目的是根据现有的数据构建线性回归模型，使之能够用于对未来趋势的预测。

如果变量之间存在某种线性的相关关系，则可以用线性回归模型来表示：

$$y=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p+\varepsilon$$

这里，$x_1,x_2,\cdots,x_p$ 称为自变量（解释变量）；y 称为因变量（被解释变量）；ε 称为误差项，是一个随机变量，反映了其他随机因素对 y 的影响。

在一个线性回归模型中，如果只有一个自变量，则称为一元线性回归模型；如果存在多个自变量，则称为多元线性回归模型。

对于误差项 ε，有 3 个基本的假定：

① 误差项 ε 是一个期望值为零的随机变量，即 $E(\varepsilon)=0$。也就是说，对于给定的 $x_1,x_2,\cdots,x_p$ 的值，y 的期望值为：$E(y)=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p$，该式也称为多元回归方程。

② 对于自变量 $x_1,x_2,\cdots,x_p$ 的所有值，ε 的方差 σ^2 都相同，即每个样本的误差项的方差均为 σ^2。

③ 误差项 ε 服从正态分布，即 $\varepsilon \sim N(0,\sigma^2)$，且每个样本的误差项相互独立。正态性意味着因变量 y 也是一个服从正态分布的随机变量；独立性则意味着每个样本的误差项是不相关的。

由于回归方程中的参数 β_0、β_1、…、β_p 是总体回归参数，往往是未知的。这就需要利用样本数据去估计它们。当用样本统计量 $\hat{\beta}_0$、$\hat{\beta}_1$、…、$\hat{\beta}_p$ 去估计回归方程中的未知参数 β_0、β_1、…、β_p 时，就得到了估计的多元回归方程：

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

其中，$\hat{y}$ 是因变量 y 的估计值；$\hat{\beta}_0$、$\hat{\beta}_1$、…、$\hat{\beta}_p$ 分别为参数 β_0、β_1、…、β_p 的估计值。$\hat{\beta}_1$、…、$\hat{\beta}_p$ 称为偏回归系数，$\hat{\beta}_i$（i=1,2,…,p）的含义是，当其他自变量不变时，x_i 每变动一个单位导致因变量 y 的平均变动量。

因此，回归分析首先要根据样本数据求出 $\hat{\beta}_0$、$\hat{\beta}_1$、…、$\hat{\beta}_p$ 的值，即参数估计，从而建立回归模型，然后还要对模型进行评估和检验。

2．参数估计

最常用的参数估计方法是最小二乘法。

对于第 i 个样本，变量 y 的估计值可以表示为：

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

把样本 i 的变量 y 的实际观测值 y_i 与估计值 $\hat{y}_i$ 之间的差称为残差，即：

$$e_i = y_i - \hat{y}_i$$

为了使得求出的回归方程能够最好地拟合所有的样本，最后求出的估计参数必须能够让所有样本的残差的平方和尽可能的小，这正是最小二乘法的基本原理，即：

$$Q = \sum (y_i - \hat{y}_i)^2 = \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_2 - \cdots - \hat{\beta}_p x_p)^2 = \text{最小}$$

根据微积分的极值定理，对 Q 求相应于各估计参数的偏导数并令其等于零，这样构成一个方程组，便可求出各估计参数：

$$\begin{cases} \dfrac{\partial Q}{\partial \beta_0}\Big|_{\beta_0=\hat{\beta}_0} = 0 \\ \dfrac{\partial Q}{\partial \beta_i}\Big|_{\beta_i=\hat{\beta}_i} = 0 \qquad (i = 1,2,\cdots,p) \end{cases}$$

例如对于一元回归方程 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$，用最小二乘法可求出其估计参数：

$$\begin{cases} \hat{\beta}_1 = \dfrac{n\sum\limits_{i=1}^{n} x_i y_i - \sum\limits_{i=1}^{n} x_i \sum\limits_{i=1}^{n} y_i}{n\sum\limits_{i=1}^{n} x_i^2 - (\sum\limits_{i=1}^{n} x_i)^2} \\ \hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \end{cases}$$

其中，n 是样本数量，$\overline{y}$ 是变量 y 的均值，$\overline{x}$ 是变量 x 的均值。

3. 回归方程的拟合优度

估计的回归方程在一定程度上描述了因变量与自变量之间的数量关系，根据这一方程，可以根据自变量的取值来预测因变量的取值。但预测的精度如何则取决于回归方程对观测数据的拟合优度（Goodness of Fit），或者说，拟合优度表示了所有的观测样本究竟在多大程度上满足估计的回归方程。

（1）判定系数

例如一个一元回归方程表示了一条直线，如果所有的观测样本点都紧密围绕在回归直线周围，就说明该回归方程的拟合优度很好。在统计学中，回归方程的拟合优度是用"判定系数"（Coefficient of Determination）来度量的。

由于自变量的取值不同，以及其他随机的因素，导致了每个样本的因变量 y 的值会有所不同，把这种波动称为"变差"。对于一个具体的样本来说，其变差的大小可以用实际观测值 y 与其均值 $\overline{y}$ 之差 $(y-\overline{y})$ 来表示。那么，n 个样本的总变差可由所有变差的平方和来表示，称为"总平方和"（Total Sum of Squares），记做 TSS：

$$\text{TSS}=\sum_{i=1}^{n}(y_i-\overline{y})^2$$

可以证明，

$$\sum_{i=1}^{n}(y_i-\overline{y})^2=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2+\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2$$

其中，$\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2$ 反映了 y 的总变差中由于各自变量与 y 之间的线性关系引起的 y 的变化部分，它是可以由回归方程来解释的变差部分，称为"回归平方和"（Explained Sum of Squares），记做 ESS。另一部分 $\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$ 是各样本实际观测值与回归值之间的残差 $(y_i-\hat{y}_i)$ 平方和，它是由其他随机因素导致的 y 的变差，是不能由回归方程来解释的变差部分，称为"残差平方和"（Residual Sum of Squares），记做 RSS。

因此有：TSS=RSS+ESS。

显然，回归方程拟合程度的大小，就看 ESS 在 TSS 中所占的比例大小，将这个比例定义为"判定系数"，记做 R^2：

$$R^2=\frac{\text{ESS}}{\text{TSS}}=\frac{\sum_{i=1}^{n}(\hat{y}_i-\overline{y})^2}{\sum_{i=1}^{n}(y_i-\overline{y})^2}\text{，或者 }R^2=1-\frac{\text{RSS}}{\text{TSS}}=1-\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\overline{y})^2}$$

R^2 的取值范围是[0,1]。R^2 越接近于 1，表示拟合程度越好；反之，R^2 越接近于 0，表示拟合程度越差。

（2）修正的判定系数

在应用过程中，人们发现随着模型中自变量的增多，判定系数 R^2 的值往往会变大，从而增加了模型的解释功能，这一事实已经在理论上得到了证实。当增加自变量时，会使预测误差变得较小，从而减少残差平方和 RSS，那么 ESS 就会变大，从而使 R^2 变大。这就给人们一种错觉：要是模型拟合得好，就必须增加自变量。但是，如果模型中增加

一个自变量，即使这个自变量在统计上并不显著，R^2 也会变大。因此，为了避免增加自变量而高估了 R^2，统计学家提出用样本容量 n 和自变量的个数 p 去修正 R^2，计算出修正的判定系数 $\overline{R}^2$：

$$\overline{R}^2 = 1 - \frac{n-1}{n-p-1}(1-R^2)$$

$\overline{R}^2$ 的解释与 R^2 类似，不同的是，$\overline{R}^2$ 同时考虑了样本容量 n 和模型中参数的个数 p 的影响，这就使得 $\overline{R}^2$ 的值永远小于 R^2，而且 $\overline{R}^2$ 的值不会由于模型中自变量数量的增加而越来越接近 1。因此，在多元回归分析中，通常用修正的判定系数。

需要注意的是，修正的判定系数 $\overline{R}^2$ 有可能为负值，在这种情况下，使用 $\overline{R}^2$ 将失去意义，此时做 $\overline{R}^2=0$ 处理。因此，$\overline{R}^2$ 只适用于因变量 y 与自变量 $x_1,x_2,\cdots,x_p$ 的整体相关程度比较高的情况。

另外，R^2 的平方根称为复相关系数，它度量了因变量同 p 个自变量的相关程度。

4. 显著性检验

对回归方程的显著性检验包括两个方面：线性关系检验和回归参数检验。

（1）线性关系的显著性检验

线性关系检验也称总体显著性检验，主要是检验因变量同多个自变量的线性关系是否显著，即检验所有自变量对因变量的“总体影响”是否显著。

线性关系显著性检验的具体步骤为：

第一步：提出假设。

$$H_0\text{：} \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

$$H_1\text{：} \beta_1, \beta_2, \cdots, \beta_p \text{ 至少有一个不等于 } 0$$

第二步：计算统计量 F。

由于 y_i 服从正态分布，根据数理统计学中的定义，y_i 的一组样本的平方和服从 χ^2 分布，所以有：

$$\text{ESS} = \sum(\hat{y}_i - \overline{y})^2 \sim \chi^2(p)$$

$$\text{RSS} = \sum(y_i - \hat{y}_i)^2 \sim \chi^2(n-p-1)$$

即回归平方和、残差平方和分布服从自由度为 p 和自由度为$(n-p-1)$的 χ^2 分布。那么根据数理统计学中的定义，在假设 H_0 成立的条件下，统计量 $F = \dfrac{\text{ESS}/p}{\text{RSS}/(n-p-1)}$ 服从分子自由度为 p 和分母自由度为$(n-p-1)$的 F 分布。

根据 ESS 和 RSS，计算出统计量 F 的值。

第三步：做出统计决策。

给定显著性水平 $\alpha=0.05$，根据分子自由度 p 和分母自由度为$(n-p-1)$，查 F 分布表得到 F_α。如果 $F > F_\alpha$，则拒绝原假设，说明回归方程线性关系显著；如果 $F \leqslant F_\alpha$，则接受原假设，说明回归方程线性关系不显著。在利用计算机软件进行计算时，也可以根据统计量 F 和两个自由度，求出显著性水平 P 值，如果 P 值 $< \alpha$，则拒绝原假设；如果 P 值 $\geqslant \alpha$，则接受原假设。

（2）回归参数的显著性检验

如果回归方程通过了上面的 F 检验，则表示方程中的所有自变量对因变量的“总体影响”是显著的，但这并不意味着每一个自变量对因变量都有重要影响，或者说并不是每个自变量的单独影响都是显著的。事实上，在上面的线性关系显著性检验中，只要有一个自变量同因变量的线性关系显著，F 检验就可以通过。

所以，回归参数的显著性检验就是对每个回归参数分布进行单独的检验，它主要用于检验每个自变量对因变量的影响是否显著。如果某个自变量没有通过检验，就意味着这个自变量对因变量的影响不显著，也许就没有必要将这个自变量放入回归模型中了。

回归参数显著性检验的具体步骤为：

第一步：提出假设。对于任意参数 β_i（i=1,2,…,p），有

$$H_0：\beta_i = 0$$

$$H_1：\beta_i \neq 0$$

第二步：计算统计量 t_i。

构造统计量 $t_i = \dfrac{\hat{\beta}_i}{s_{\hat{\beta}_i}}$。其中，$s_{\hat{\beta}_i}$ 是回归参数 $\hat{\beta}_i$ 的抽样分布的标准差，即：

$$s_{\hat{\beta}_i} = \frac{s_y}{\sqrt{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2}}$$

这里，s_y 是回归模型 $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$ 中误差项 ε 的方差 σ^2 的一个估计值：$s_y = \sqrt{\dfrac{\sum (y_i - \hat{y}_i)^2}{n-p-1}} = \sqrt{\dfrac{\text{RSS}}{n-p-1}}$。

可以证明，统计量 t_i 服从自由度为$(n-p-1)$的 t 分布，即 $t_i \sim t(n-p-1)$。

第三步：做出统计决策。

给定显著性水平 α=0.05，根据自由度$(n-p-1)$，查 t 分布表得到 $t_{\alpha/2}$。如果 $|t_i| > t_{\alpha/2}$，则拒绝原假设，说明自变量 x_i 对因变量 y 的影响显著；如果 $|t_i| \leqslant t_{\alpha/2}$，则接受原假设，说明自变量 x_i 对因变量 y 的影响不显著。在利用计算机软件进行计算时，也可以根据统计量 t_i 和自由度 $n-p-1$，求出显著性水平 P 值，如果 P 值 $< \alpha$，则拒绝原假设；如果 P 值 $\geqslant \alpha$，则接受原假设。

5. 标准化系数

在多元线性回归方程中，由于各自变量的单位不同，得到的回归系数也就有不同的量纲。因此，回归系数的大小只能表明自变量与因变量在数量上的关系，而不能表示各自变量在回归方程中的重要性。要比较各个自变量的重要性，必须消除单位的影响。为此，在做线性回归时需要对变量值做标准化变换，即变量值减去其均值并除以其标准差的估计，由此得到的回归系数被称为**标准化系数**。

标准化的回归模型为：

$$\frac{y_i-\overline{y}}{\sigma_y}=\beta_1'\frac{x_{1i}-\overline{x}_1}{\sigma_{x_1}}+\beta_2'\frac{x_{2i}-\overline{x}_2}{\sigma_{x_2}}+\cdots+\beta_p'\frac{x_{pi}-\overline{x}_p}{\sigma_{x_p}}+\varepsilon_i$$

其中，σ_y 是因变量 y 的标准差：$\sigma_y=\sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(y_j-\overline{y})^2}$，

σ_{x_i} 是自变量 x_i 的标准差：$\sigma_{x_i}=\sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_{ij}-\overline{x}_i)^2}$，$n$ 为样本数量。

因标准化系数没有单位，可用其绝对值大小来说明多元回归模型中各自变量的相对重要性。标准化系数的含义是当自变量增加一个单位时，因变量增加或减少的单位数，它与原来未标准化的多元回归系数的关系为：

$$\hat{\beta}_i'=\hat{\beta}_i\frac{\sigma_{x_i}}{\sigma_y}，\quad i=1,2,\cdots,p$$

可以看出，标准化系数不仅与自变量的回归系数有关，而且与这个自变量的波动程度（自变量的标准差）有关。

7.1.2 在 Clementine 中应用线性回归

在 Clementine 中，一元或多元线性回归分析由回归节点来完成，它采用最小二乘法来根据样本数据建立回归方程。注意，在回归模型中只能使用连续型字段。必须有且仅有一个目标字段（输出字段，因变量），但可以有一个或多个预测变量（输入字段，自变量）。方向为“两者”（双向）或“无”的字段将被忽略。

本小节根据数据样本集来建立线性回归模型。数据样本集如表 7-1 所示。

表 7-1 训练数据样本

样品	X1	X2	X3	Y
1	0.4	53	158	64
2	0.4	23	163	60
3	3.1	19	37	71
4	0.6	34	157	61
5	4.7	24	59	54
6	1.7	65	123	77
7	9.4	44	46	81
8	10.1	31	117	93
9	11.6	29	173	93
10	12.6	58	112	51
11	10.9	37	111	76
12	23.1	46	114	96
13	23.1	50	134	77
14	21.6	44	73	93
15	23.1	56	168	95
16	1.9	36	143	54
17	26.8	58	202	168
18	29.9	51	124	99

数据集包含 18 个样本，4 个变量，其中变量 Y 是因变量，$X1$、$X2$ 和 $X3$ 是自变量。数据集存放在 Excel 文件“线性回归样本.xls”中。

完整的数据流如图 7.1 所示。

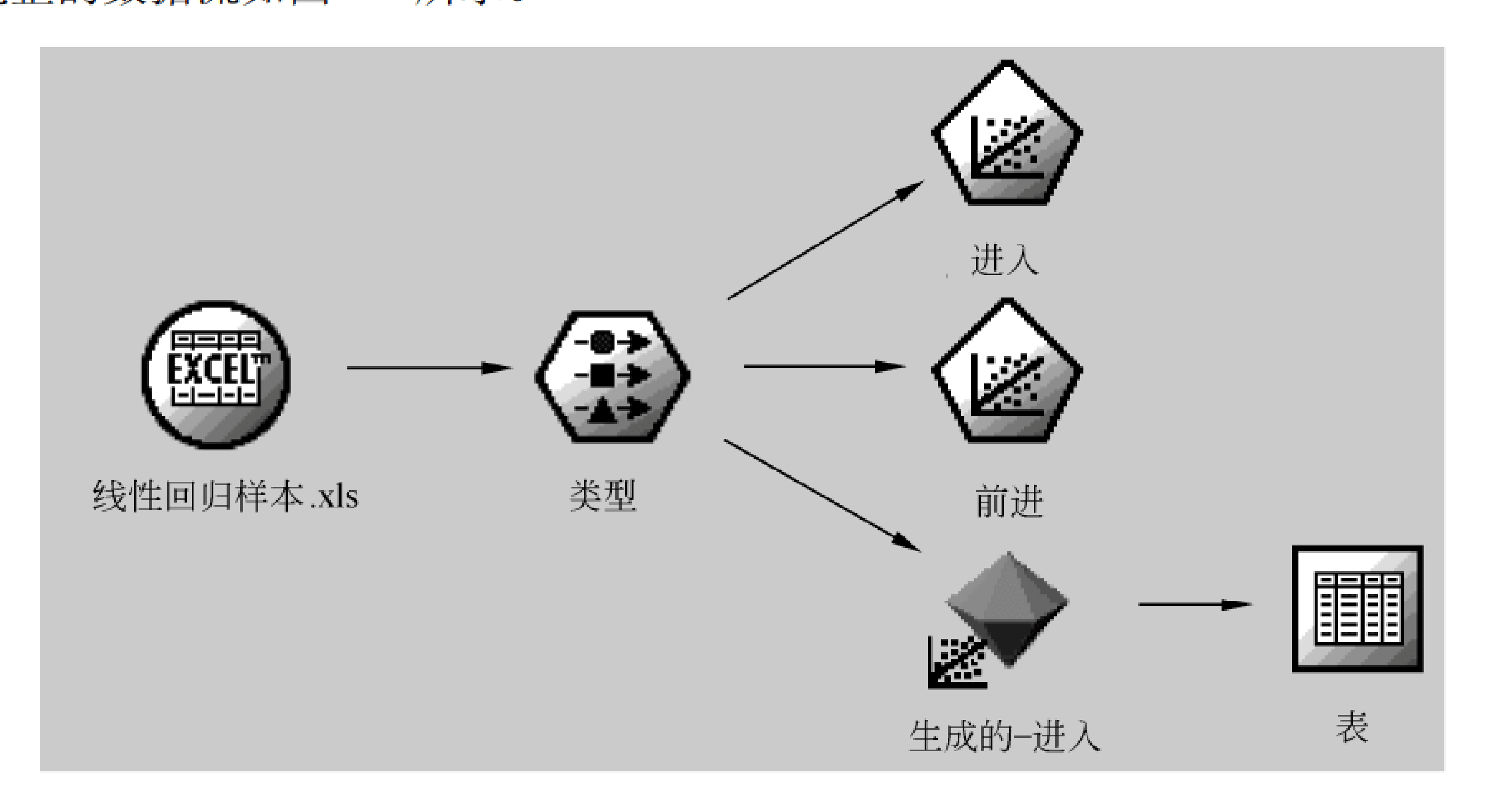


图 7.1　线性回归数据流

首先，将“数据源”中的 Excel 节点添加到数据流区域，并将“线性回归样本.xls”文件加载到该节点。

向数据流中添加“类型”节点，并建立从“线性回归样本.xls”节点到“类型”节点的连接。打开“类型”节点的编辑窗口，把“样品”字段的方向设置为“无”，即该字段不参与建模。把 $X1$、$X2$ 和 $X3$ 的方向设置为“输入”，成为自变量。把 Y 的方向设置为“输出”，成为因变量。然后单击“读取值”按钮，如图 7.2 所示。



图 7.2　设置“类型”节点

1．使用“进入法”建模

向数据流中添加“回归”节点，建立从“类型”节点到“回归”节点的连接，然后对“回归”节点进行设置。

首先，在节点编辑窗口的“注解”标签下，将该节点的名称“自定义”为“进入”。

切换到“模型”标签下，设置如图 7.3 所示。



图 7.3　建模的基本设置

在“模型名称”的选项中选择“自定义”，并在文本框中输入“生成的-进入”，这是设置即将建立的线性回归模型的名称。

单击“方法”（method）下拉列表框，可以看到，有 4 种建模的方法可供选择：

（1）**进入法（Enter）**：这是默认的方法，该方法将所有输入字段直接纳入方程。构建模型时不进行字段选择。本例在这里选择“进入法”。

（2）**逐步法（Stepwise）**：逐步字段选择法就是分步构建方程。初始模型是可能的最简单模型，方程中没有任何输入字段。每个步骤会对尚未添加到模型中的输入字段进行评估，如果其中的最佳输入字段对模型的预测能力有显著作用，则会添加该字段。此外，还会重新评估当前包含在模型中的输入字段，以确定能否在不对模型的功能造成重大减损的情况下删除其中的任何字段。如果可以，则会将其删除。然后重复此过程，添加并/或删除其他字段。当无法再添加任何字段来改进模型，且无法再删除任何字段而不对模型功能造成减损时，最终模型便已生成。

（3）**后退法（Backwards）**：后退字段选择法与分步构建模型的逐步法类似。但采用这种方法时，初始模型包含作为预测变量的所有输入字段，只能从模型中删除字段。对模型贡献较小的输入字段将被逐一删除，直到无法再删除任何字段而不对模型功能造成重大损害，从而生成最终模型。

（4）**前进法（Forwards）**：前进法与后退法是相反的。采用这种方法，初始模型是不包含任何输入字段的最简单模型，只能向模型中添加字段。每个步骤会对尚未纳入到模型中的输入字段进行检验，看它们对模型的改进起多大作用，然后将其中最佳的字段添加到模型中。当无法再添加任何字段或最佳备选字段无法对模型产生足够的改进时，最终模型便已生成。

勾选“**将常量纳入方程式**”（**Include Constant In Equation**）复选框，此选项用于确定结果方程式是否将包含常数项。在大多数情况下，应将此选项保持为选中状态。如果已经明确地知道模型中的常数项为 0，则不勾选此选项。

切换到“专家”标签下，在“模式”的选项中选择“专家”单选按钮，即可对建模进行高级设置，如图 7.4 所示。



图 7.4　建模的高级设置

缺失值（Missing Values）：默认勾选“仅使用完整记录”复选框，回归节点将仅使用对于模型中使用的所有字段均具有有效值的记录。如果有很多缺失数据，可以取消选中“仅使用完整记录”复选框，Clementine 将尝试使用尽可能多的信息来估计回归模型，包括其中一些字段具有缺失值的记录，但在某些情形下，以这种方式使用不完整记录可能会在估计回归方程时导致计算问题。

单击“输出…”按钮，弹出“线性回归：高级输出选项”对话框，可以选择在生成的模型中输出哪些统计信息，如图 7.5 所示。

其中：

模型拟合（Model Fit）：模型拟合概要信息，包括拟合优度 R^2。

回归系数（Regression Coefficients）：回归系数的相关统计量。

R 平方改变量（R squared Change）：R^2 在逐步、前进和后退估计法的每个步骤中的改变量。

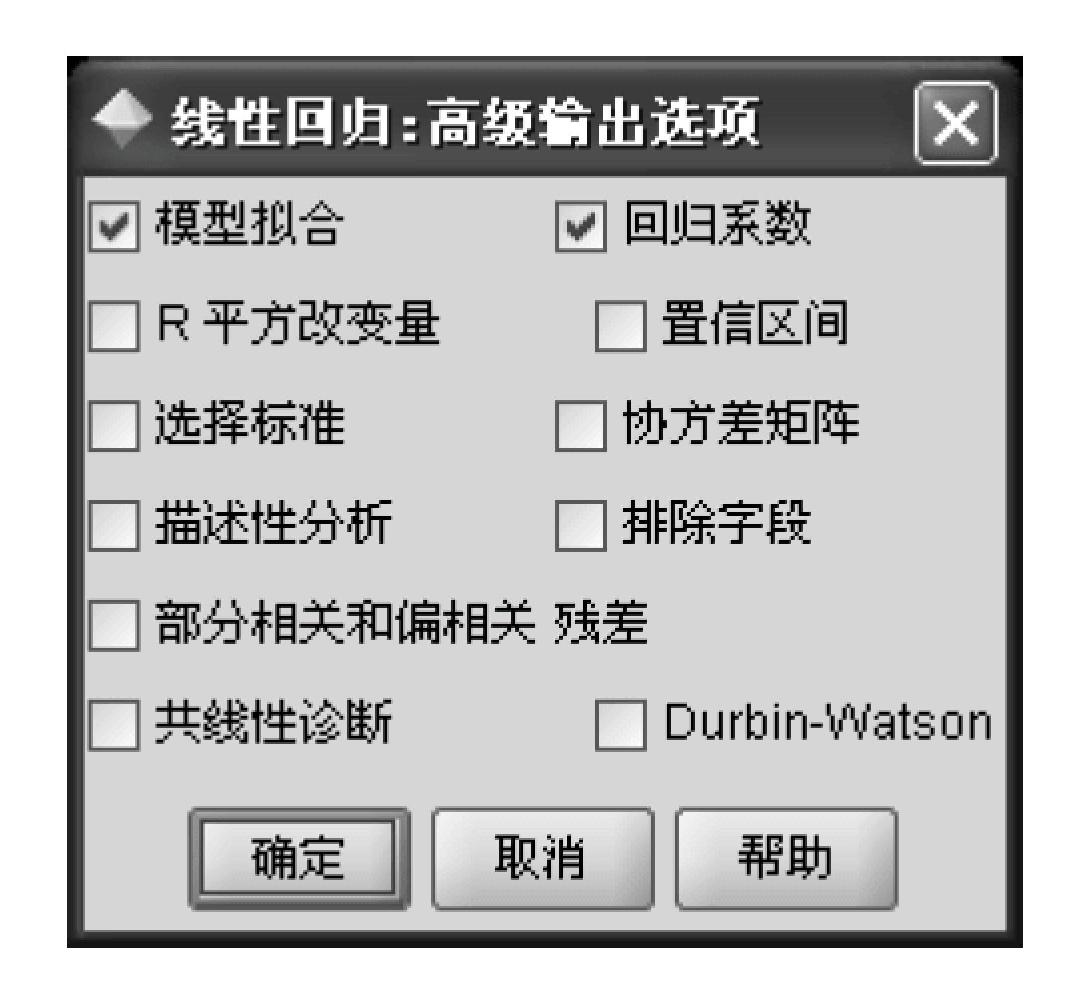


图 7.5 “线性回归：高级输出选项”对话框

置信区间（Confidence Interval）： 方程式中每个回归系数的 95%置信区间。

选择标准（Selection Criteria）： 用于针对模型的每个步骤估计模型的信息内容（以帮助评估模型改进情况）的统计量。这些统计量包括 AIC 信息准则（Akaike Information Criterion）、Amemiya 预测标准（Amemiya’s Prediction Criterion）、Mallows 预测标准（Mallows’ Prediction Criterion）和 SBC 标准（Schwarz Bayesian Criterion）。

协方差矩阵（Covariance Matrix）： 输入字段的协方差矩阵。

描述性分析（Descriptives）： 有关输入和输出字段的基本描述性统计量。

排除字段（Exclude Fields）： 曾被考虑包括在模型中但根据所用的选择方法（逐步法、后退法等）最终被拒绝的字段的相关统计量。

部分相关和偏相关（Part And Partial Correlations）： 帮助确定各个输入字段对模型的重要性和独有贡献的统计量。

残差（Residuals）： 预测值与实际值之间的差异。

共线性诊断（Collinearity Diagnostics）： 用于找出因多余输入字段引起的问题的统计量。

Durbin-Watson： 自相关的 Durbin-Watson 检验。检验记录顺序对回归模型的影响。

以上设置完成后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示“生成的-进入”模型。

2. 自动选择变量建模

当采用逐步法、后退法或者前进法建模时，将根据某些设置来对输入变量进行筛选，选择那些对因变量影响显著的自变量来建模。在建模的过程中，有可能将模型中已有的变量移除，也可能将模型中没有的变量纳入到模型中来。

由用户自行设定“移除标准”值和“纳入标准”值。当要对某个自变量 x_k 进行取舍时，计算统计量 F-to-enter$_k$ 和 F-to-remove$_k$ 的值。

对于一个目前未被纳入到模型中的自变量，如果其 F-to-enter$_k$ 大于“纳入标准”值，则将其纳入到模型中来。

对于一个目前已被纳入到模型中的自变量，如果其 F-to-remove$_k$ 小于“移除标准”值，则将其从模型中移除。

下面介绍 F-to-enter$_k$ 和 F-to-remove$_k$ 的计算方法：

对于一个回归模型：

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \varepsilon_i$$

可以计算出任意两个变量之间的相关系数并构成相关系数矩阵：

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & \cdots & r_{1p} & r_{1y} \\ r_{21} & \cdots & r_{2p} & r_{2y} \\ \vdots & \ddots & & \vdots \\ r_{y1} & \cdots & r_{yp} & r_{yy} \end{bmatrix}$$

例如 r_{12} 表示变量 x_1 和 x_2 的相关系数，r_{1y} 表示变量 x_1 和 y 的相关系数。

对于任意两个变量 x 和 y，它们的相关系数为 $r_{xy} = \dfrac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{(x_i - \overline{x})^2 \sum (y_i - \overline{y})^2}}$。

对于尚未纳入到模型中的变量 x_k，如果同时满足以下 3 个条件，它将被纳入到模型中：

（1）$r_{kk}>t$，这里 t 是“异常值容忍度”（Singularity Tolerance），由用户自行设定，默认值为 0.0001。

（2）对于模型中所有已有的变量 x_j，$\left(r_{jj} - \dfrac{r_{jk} r_{kj}}{r_{kk}} \right) t \leqslant 1$。这个条件使得 x_k 的加入不会让其他已有的变量的异常值容忍度下降到一个不能接受的程度。

（3）$\text{F - to - enter}_k = \dfrac{(C - p^* - 1)V_k}{r_{yy} - V_k}$ 大于“纳入标准”值。这里，C 是有效样本的数量，p^*是回归模型中系数的数量，$V_k = \dfrac{r_{yk} r_{ky}}{r_{kk}}$。由于这里的统计量 F-to-enter$_k$ 服从自由度为 1 和 $(C - p^* - 1)$ 的 F 分布，因此也可以把“纳入标准”设置为概率阈值，根据 F-to-enter$_k$ 求出相应的 p 值，如果 p 值小于概率阈值，则将变量纳入。

对于已经被纳入到模型中的变量 x_k，如果 $\text{F - to - remove}_k = \dfrac{(C - p^*)|V_k|}{r_{yy}}$ 小于“移除”标准值，该变量将被从模型中移除。由于这里的统计量 F-to-remove$_k$ 服从自由度为 1 和 $(C - p^*)$ 的 F 分布，因此也可以把“纳入标准”设置为概率阈值，根据 F-to-enter$_k$ 求出相应的 p 值，如果 p 值大于概率阈值，则将变量移除。

下面以前进法为例来说明这一过程。

向数据流中添加“回归”节点，建立从“类型”节点到“回归”节点的连接，然后对“回归”节点进行设置。在节点编辑窗口的“注解”标签下，将该节点的名称“自定义”为“前进”。

切换到“模型”标签下，将建模方法设置为“前进法”。在“模型名称”的选项中选择“自定义”，并在文本框中输入“生成的-前进”，这是设置即将建立的线性回归模型的名称。

切换到“专家”标签下，可以对“异常值容忍度”（Singularity Tolerance）进行设置，它用于指定一个字段独立于模型中其他字段的最小方差比率，这就是上面提到的参数 t，默认值为 0.0001。

单击“步进…”按钮可以打开“线性回归：步进标准”对话框，如图 7.6 所示。



图 7.6 “线性回归：步进标准”对话框

选择两个步进标准中的一个，并根据需要更改阈值，即根据 F 统计量的大小或者 p 值来确定是否将自变量纳入模型中来，或者将自变量从模型中移除。显然，这两种标准之间存在互逆关系。字段对模型越重要，p 值就越小，而 F 值则越大。

使用 F 的概率：即设置上面提到的概率阈值，包括纳入概率阈值（采用逐步法和前进法时需要设置）和移除概率阈值（采用逐步法和后退法时需设置）。

使用 F 值：即设置 F-to-enter$_k$ 和 F-to-remove$_k$ 的阈值。

以上设置完成后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示“生成的-前进”模型。

3. 浏览模型

右击管理器窗口的“模型”标签下的“生成的-进入”模型，选择“浏览”，可以打开模型的浏览窗口，该窗口显示了采用进入法所生成回归模型的信息细节，在“模型”标签下，显示了变量重要性；在“汇总”标签下，显示了本次建模的结果和信息概要，如图 7.7 所示。



图 7.7 “生成的-进入”模型汇总信息

从“分析”中显示的估计参数的值，可以得到最终的回归模型为：

$$y = 43.65 + 1.785x_1 - 0.0834x_2 + 0.1611x_3 + \varepsilon$$

在窗口的“高级”标签下，通过“模型摘要”、ANOVA 和“系数”等表格，显示了本次建模过程中的细节信息。

“模型摘要”如图 7.8 所示。

模型	R	R 方	调整的 R 方	估计的标准差
1	.741(a)	.549	.453	19.97051
a. 预测变量:(常量), X3, X1, X2。				

图 7.8　模型摘要

从中可以看出，判定系数 R^2=0.549；调整的判定系数为 0.453；估计的标准差 s_y = 19.97051。

ANOVA 如图 7.9 所示。

模型		平方和	df	均方	F	显著性
1	回归	6806.111	3	2268.704	5.689	.009(b)
	残差	5583.500	14	398.821		
	合计	12389.611	17			
a. 因变量: Y						
b. 预测变量:(常量), X3, X1, X2。						

图 7.9　ANOVA

可以看出，回归平方和 ESS =6806.111，自由度为 3，均方为 6806.111/3=2268.704；残差平方和 RSS =5583.5，自由度为 14，均方为 5583.5/14=398.821；总平方和 TSS= RSS+ESS =12389.611；线性关系显著性检验的统计量 F=5.689，对应的 p 值为 0.009，小于 0.05，说明回归方程的线性关系显著。

“系数”表格如图 7.10 所示。

模型		非标准化系数		标准化系数	t	显著性
		B	标准误	Beta		
1	(常量)	43.652	18.010		2.424	.029
	X1	1.785	.538	.671	3.319	.005
	X2	-8.34E-002	.418	-.042	-.200	.845
	X3	.161	.112	.273	1.443	.171
a. 因变量: Y						

图 7.10　估计参数

图 7.10 中列出了参数的估计量及相应的标准误差，参数估计量的标准误差越小，则估计值与真实值的误差就越小。从回归参数的显著性检验来看，X_2 和 X_3 对因变量 Y 的影响不显著，它们的 t 统计量所对应的 p 值均大于 0.05。

所以，在采用“前进法”生成的模型“生成的-前进”中，通过浏览该模型可以发现，生成的模型中只保留了自变量 X_1。

对于所有生成的回归模型，可以用于根据自变量的值来预测因变量的值。例如，将“生成的-进入”模型拖入数据流区域，并建立从“类型”节点到“生成的-进入”节点的连接，然后在“生成的-进入”节点后面添加“表”节点，执行该“表”节点，即可生成预测结果，如图 7.11 所示。

表（6 个字段，18 条记录） #1

文件(F) 编辑(E) 生成(G)

	样品	X1	X2	X3	Y	$E-Y
1	1.000	0.400	53.000	158.000	64.000	65.405
2	2.000	0.400	23.000	163.000	60.000	68.713
3	3.000	3.100	19.000	37.000	71.000	53.562
4	4.000	0.600	34.000	157.000	61.000	67.185
5	5.000	4.700	24.000	59.000	54.000	59.546
6	6.000	1.700	65.000	123.000	77.000	61.085
7	7.000	9.400	44.000	46.000	81.000	64.172
8	8.000	10.100	31.000	117.000	93.000	77.946

表 注解

确定(O)

图 7.11 基于回归模型的预测结果

在$E-Y 一列中，显示了基于回归模型的预测结果。

7.2 二项 Logistic 回归

7.2.1 二项 Logistic 回归的基本原理

1. 基本概念

线性回归模型的一个局限性是要求因变量和自变量都是连续型变量（定距变量、定比变量），而不能是离散型变量（定序变量、定类变量）。但是在许多实际问题中，经常出现因变量或自变量是离散型变量（分类变量）的情况，例如要研究因变量“购买汽车”，这个因变量就是一个离散型变量，它只有两个取值，即“买”和“不买”，这个因变量可能与自变量“性别”相关，这个自变量也是一个离散型变量，它的两个取值是“男”和

“女”。显然，当遇到此类问题的时候，线性回归方法就不再适用了。这时可以采用 Logistic 回归分析方法。

Logistic 回归分析根据因变量取值类别不同，又可以分为二项 Logistic 回归（二分类 Logistic 回归，Binary Logistic 回归）分析和多项 Logistic 回归（多分类 Logistic 回归，Multinomial Logistic 回归）分析，二项 Logistic 回归模型中因变量只能取两个值，分别用 1 和 0 来表示，而多项 Logistic 回归模型中因变量可以取多个值。本节只讨论二项 Logistic 回归。

对于一个取值为 0 和 1 的因变量 y，可以采用多种方法来对其进行分析。通常以 p 表示 y=1 的概率（事件发生的概率），1–p 表示 y=0 的概率（事件不发生的概率），并把 p 看做自变量 x_i 的线性函数，即 $p = P(y=1) = F(\beta_i x_i)$，$i$=1,2,⋯,$k$。不同形式的 $F(\bullet)$，就有不同形式的模型，最简单的莫过于使 $F(\bullet)$为一线性函数，即：

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon$$

因此，所研究的因变量由 y 变成了 p，即要研究因变量 p 与自变量 x_i 之间的线性关系。但是，由于 p 的值一定在区间[0,1]内，而且当 p 接近于 0 或 1 时，自变量即使有很大变化，p 的值也不可能变化很大，所以对上式直接用普通最小二乘法进行参数估计是行不通的。

从数学上看，函数 p 对 x_i 的变化在 p=0 或 p=1 的附近是不敏感的、缓慢的，且非线性的程度较高。于是要寻求一个 p 的函数 $\theta(p)$，使得它在 p=0 或 p=1 附近时变化幅度较大，而函数的形式又不是很复杂。因此，引入 p 的 Logistic 变换（或称为 p 的 Logit 变换），即：

$$\theta(p) = \mathrm{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

其中，$\frac{p}{1-p}$称为“发生比”或者“相对风险”。

经过 Logit 变换之后，就可以利用一般线性回归模型建立因变量 $\theta(p)$ 与自变量 x_i 之间的线性关系模型：

$$\theta(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon$$

现在，首先要根据样本数据，对上式中的参数 β_i 进行参数估计，从而建立 Logistic 回归模型，然后还要对模型进行评估和检验。最后，应用所建立的模型，可以根据回归方程，预测给定样本的因变量 y=1 的概率 p 进行预测。p 值的计算如下：

根据回归方程 $\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$，有 $\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k}$，从而可推出 p 值为：$p = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k}}$。

2．参数估计

由于 Logistic 回归模型的残差不再服从正态分布，而是二值离散型分布，所以采用极大似然估计法对模型的参数进行估计。

首先建立似然函数 L：

$$L=\prod_{i=1}^{n}\frac{e^{\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k}}{1+e^{\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k}}$$ ，其中，n 是样本数量

对上式两边取对数得到：

$$\ln(L)=\sum_{i=1}^{n}\ln\frac{e^{\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k}}{1+e^{\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k}}$$

然后分别对 $\beta_0,\beta_1,\cdots,\beta_k$ 求偏导并令：

$$\frac{\partial\ln L}{\partial\beta_0}=0\text{，}\quad\frac{\partial\ln L}{\partial\beta_1}=0\text{，}\quad\cdots\text{，}\quad\frac{\partial\ln L}{\partial\beta_k}=0$$

通过上述方程组求解，即可求出 $\beta_0,\beta_1,\cdots,\beta_k$。

3．二项 Logistic 回归分析中的虚拟变量

在二项 Logistic 回归分析中，如果有些自变量为定类或者定序的离散型变量，则必须将其转化成虚拟变量。当某一自变量有 m 种分类（m 个取值）时，需要设立 m–1 个虚拟变量。

例如，如果 Logistic 回归方程 $\ln\left(\frac{p}{1-p}\right)=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k+\varepsilon$ 中，x_2 是一个二分变量（取值为“买”或“卖”，或者“男”或“女”），那么将其转换为虚拟变量 d_2，当 x_2 取第一种类型值时，d_2=1；当 x_2 取第二种类型值时，d_2=0。这时 Logistic 回归方程变成了：

$$\ln\left(\frac{p}{1-p}\right)=\beta_0+\beta_1x_1+\beta_2d_2+\cdots+\beta_kx_k+\varepsilon$$

再例如，如果 Logistic 回归方程中，x_2 是一个三分变量，需要对其设置两个虚拟变量 d_1 和 d_2，这时 Logistic 回归方程变成了：

$$\ln\left(\frac{p}{1-p}\right)=\beta_0+\beta_1x_1+\beta_{21}d_1+\beta_{22}d_2+\cdots+\beta_kx_k+\varepsilon$$

其中，当 x_2 取第一种类型值时，d_1=1 和 d_2=0，其回归方程为：

$$\ln\left(\frac{p_1}{1-p_1}\right)=\beta_0+\beta_1x_1+\beta_{21}+\beta_3x_3\cdots+\beta_kx_k+\varepsilon$$

当 x_2 取第二种类型值时，d_1=0 和 d_2=1，其回归方程为：

$$\ln\left(\frac{p_2}{1-p_2}\right)=\beta_0+\beta_1x_1+\beta_{22}+\beta_3x_3\cdots+\beta_kx_k+\varepsilon$$

当 x_2 取第三种类型值时，d_1=0 和 d_2=0，其回归方程为：

$$\ln\left(\frac{p_3}{1-p_3}\right)=\beta_0+\beta_1x_1+\beta_3x_3\cdots+\beta_kx_k+\varepsilon$$

4．显著性检验

对 Logistic 回归方程的显著性检验包括两个方面：线性关系检验和回归参数检验。

（1）线性关系的显著性检验

Logistic 回归方程线性关系的显著性检验的目的，是检验全体自变量与 $\ln\left(\frac{p}{1-p}\right)$ 的线性关系是否显著。

由于 Logistic 回归方程求解参数是采用最大似然估计方法，因此对回归方程的显著性也通过似然函数值来判断。似然函数值是在假设拟合模型成立的条件下能够观测到这一特定样本的概率。极大似然函数值 L 是一个在[0,1]之间的很小的数，对 L 取对数后得到的 $\ln(L)$ 必然小于 0。所以，通常将 L 取对数后再乘以–2，即–2 $\ln(L)$。–2 $\ln(L)$ 越大意味着回归模型的似然值越小，模型的拟合程度越差；–2$\ln(L)$ 越小意味着模型的似然值越大，似然值越接近于 1，拟合程度越好；如果似然值等于 1，则表示模型完全拟合了观测值。

在检验 Logistic 回归模型时，通常将回归模型与截距模型相比较。所谓截距模型是指如下形式的模型：

$$\ln\frac{p}{1-p}=\beta_0，\quad \beta_0 \text{ 为常数}$$

该模型没有引入任何自变量，它的似然值最小，是一个“不好”的模型。以截距模型作为“基准”，比较当模型中引入了自变量后新的模型与数据的拟合水平是否差别显著。如果差别越大，表示新的模型越有效。

所以，线性关系的显著性检验的步骤如下：

第一步：定义截距模型，用 L_0 表示截距模型的似然值。

第二步：对于所要检验的模型，其包含有若干个自变量，其似然值为 L。

第三步：构造对数似然比的统计量 G^2：

$$G^2=2\ln\left(\frac{L}{L_0}\right)=(-2\ln L_0)-(-2\ln L)$$

G^2 近似服从 χ^2 分布，其自由度为检验回归模型的自变量的个数 k。

第四步：提出假设：

$$H_0：\beta_1=\beta_2=\cdots=\beta_k=0$$
$$H_1：\beta_1,\beta_2,\cdots,\beta_k \text{ 至少有一个不等于 } 0$$

第五步：给出显著性水平 α。计算似然比卡方的观测值 G^2 和对应的 p 值。如果 $G^2\leqslant\chi^2_\alpha(k)$（即 p 值大于等于显著性水平 α），则接受零假设，认为目前方程中的所有回归系数同时为零，自变量全体与 $\ln\left(\frac{p}{1-p}\right)$ 之间的线性关系不显著；如果 $G^2>\chi^2_\alpha(k)$（即 p 值小于显著性水平 α），则拒绝零假设，认为目前方程中的所有回归系数不同时为零，自变量全体与 $\ln\left(\frac{p}{1-p}\right)$ 之间的线性关系显著。

（2）回归参数的显著性检验

Logistic 回归参数的显著性检验的目的是逐个检验模型中的各自变量是否与 $\ln\left(\frac{p}{1-p}\right)$ 有显著的线性关系，即对解释 $\ln\left(\frac{p}{1-p}\right)$ 是否有重要贡献。

回归参数显著性检验的具体步骤为：

第一步：提出假设。对于任意参数 β_i（i=1,2,…,k），有

$$H_0：\beta_i = 0$$
$$H_1：\beta_i \neq 0$$

第二步：计算 Wald 统计量。

Wald 统计量的数学定义是：$\text{Wald} = \left(\dfrac{\beta_i}{S_{\beta_i}}\right)^2$，其中 β_i 是回归参数，S_{β_i} 是 β_i 的标准误差。Wald 统计量服从自由度为 1 的卡方分布。

第三步：做出统计决策。

计算各自变量的 Wald 的观测值和对应的概率 p 值。如果对于某自变量，计算出的 p 值小于给定的显著性水平 α，则拒绝零假设，认为该自变量与 $\ln\left(\dfrac{p}{1-p}\right)$ 之间的关系显著，应该保留在回归方程中；如果 p 值大于等于给定的显著性水平 α，则接受零假设，认为该自变量与 $\ln\left(\dfrac{p}{1-p}\right)$ 之间的关系不显著，不应该保留在回归方程中。

5．回归方程的拟合优度检验

拟合优度表示回归方程能够解释因变量的变差的程度。如果方程可以解释因变量的较大部分变差，则说明拟合优度高，反之则拟合优度低。另外，也可以从回归方程的预测准确度来衡量其拟合优度。

常用的 Logistic 回归方程的拟合优度检验方法有以下几种。

（1）基于 Cox & Snell R^2 统计量的优度检验

Cox & Snell R^2 与一般线性回归分析中的 R^2 有相似之处，也是方程对因变量变差解释程度的反映。Cox & Snell R^2 的数学定义为：

$$\text{Cox \& Snell } R^2 = 1 - \left(\frac{L_0}{L}\right)^{\frac{2}{n}}$$

其中，L_0 是方程中只包含常数项时的似然函数值，L 是当前方程的似然函数值，n 为样本数量。

Cox & Snell R^2 的取值范围不易确定，解释时有一定困难。

（2）基于 Nagelkerke R^2 统计量的优度检验

Nagelkerke R^2 是修正的 Cox & Snell R^2，也反映了方程对因变量变差解释的程度。Nagelkerke R^2 的数学定义为：

$$\text{Nagelkerke } R^2 = \frac{\text{Cox \& Snell } R^2}{1-(L_0)^{\frac{2}{n}}}$$

Nagelkerke R^2 的取值范围在 0～1 之间。越接近于 1，说明方程的拟合优度越高，越接近于 0，说明方程的拟合优度越低。

（3）基于错分类表的优度检验

分类表是一种非常直观的评价模型的方法，它通过表格的形式展现了预测值与实际

观测值的吻合程度。分类表的一般形式如表 7-2 所示。

表 7-2　分类表

		预测值		
		0	1	正确率百分比
观测值	0	f_{11}	f_{12}	$\frac{f_{11}}{f_{11}+f_{12}}$
	1	f_{21}	f_{22}	$\frac{f_{22}}{f_{21}+f_{22}}$
		总百分比	$\frac{f_{11}+f_{22}}{f_{11}+f_{12}+f_{21}+f_{22}}$	

表 7-2 中，f_{11} 是观测值为 0 预测值也为 0 的样本数量；f_{12} 是观测值为 0 但预测值为 1 的样本数量；f_{21} 是观测值为 1 但预测值为 0 的样本数量；f_{22} 是观测值为 1 预测值也为 1 的样本数量。显然，正确率百分比越高，意味着模型越好。

7.2.2　在 Clementine 中应用 Logistic 回归

在 Clementine 中，二项 Logistic 回归分析由 Logistic 节点来完成，该节点也可以进行多项 Logistic 分析。

在利用该节点进行 Logistic 回归分析时，必须指定一个或多个预测变量（In 字段，即自变量）和唯一一个分类目标（Out 字段，即因变量）。当进行二项 Logistic 回归分析时，目标字段必须指定为标志型字段。方向为“两者”（双向）或“无”的字段将被忽略。

本小节描述一个根据数据样本集来建立 Logistic 回归模型的案例，该案例参考了 Clementine 自带的一个应用示例。某电信服务提供商希望根据客户的历史数据来分析哪些客户有可能流失，以及客户的流失与哪些因素有关。数据集存放在文件 telco.sav 中，该文件的存放路径为...Clementine12.0\Demos\telco.sav，是 Clementine 的自带文件。该数据集包括 region（地区）、tenure（服务的月数）、age（年龄）以及 churn（客户流失）等 42 个字段，且包含有 1000 个样本记录。这里，churn 字段的取值为 0 或者 1，其中“0”表示没有流失，“1”表示流失。以 churn 为因变量，首先从剩下 41 个字段中选择出最重要的若干个字段作为自变量，然后建立 Logistic 回归模型。完整的数据流如图 7.12 所示。

1. 生成模型

首先，将“数据源”中的“SPSS 文件”节点添加到数据流区域，并将 telco.sav 文件加载到该节点。

向数据流中添加“类型”节点，并建立从 telco.sav 节点到“类型”节点的连接。打

开“类型”节点的编辑窗口，将 churn 字段的类型设置为“标志”，方向设置为“输出”，成为因变量。然后单击“读取值”按钮，如图 7.13 所示。

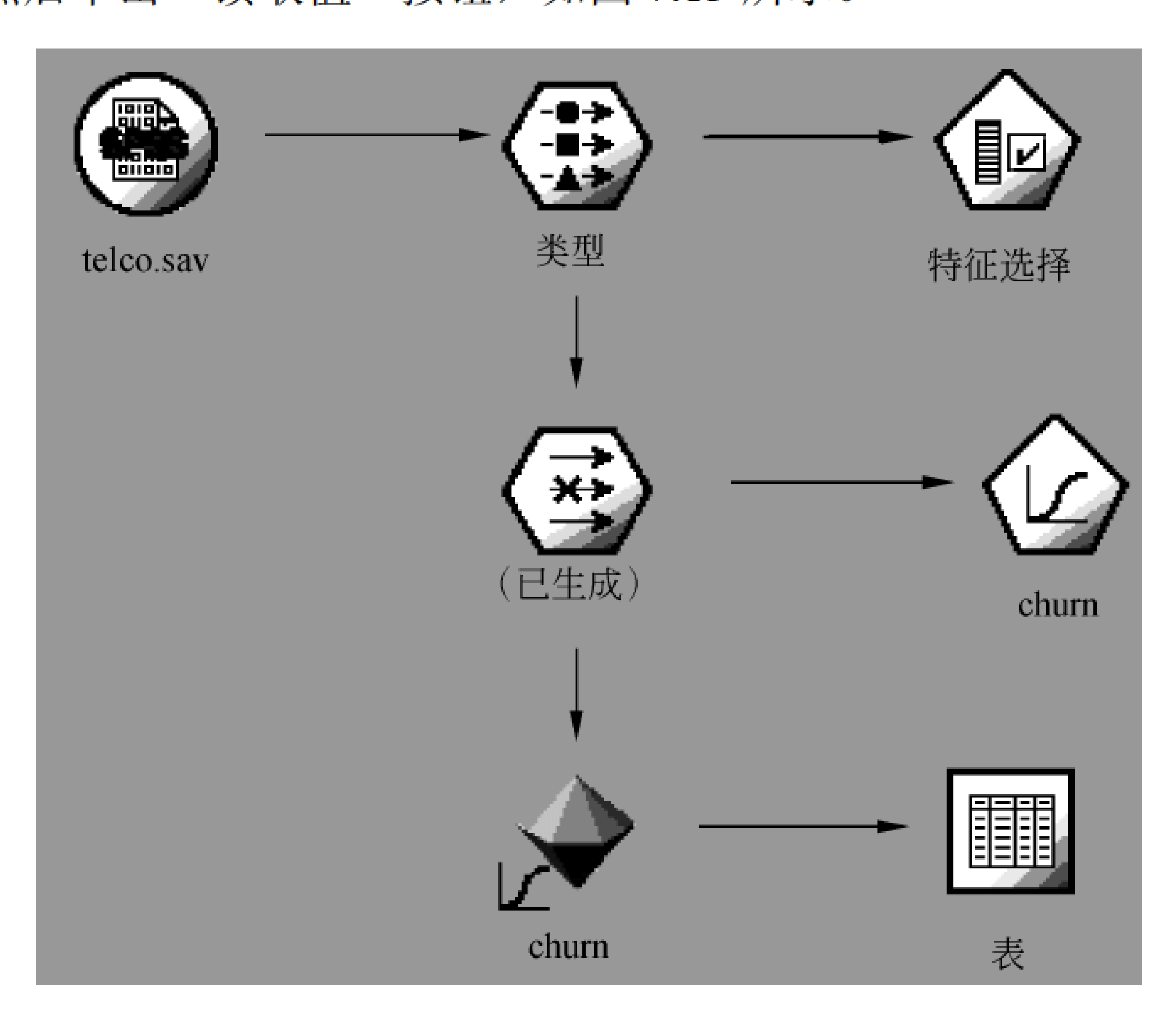


图 7.12 Logistic 回归分析数据流



图 7.13 设置“类型”节点

向数据流中添加“特征选择”节点，建立从“类型”节点到“特征选择”节点的连接，然后打开“特征选择”节点的设置窗口进行设置。

在“特征选择”节点设置窗口的“模型”标签下，“模型名称”选择“自定义”，并在文本框中输入“生成的-选择”，并对“屏幕字段”中的各指标的阈值进行相应的设置，如图 7.14 所示。



图 7.14 特征选择节点的基本设置

在“特征选择”节点设置窗口的“选项”标签中的设置如图 7.15 所示，即选择 importance 系数大于 0.999 的字段。



图 7.15 特征选择节点的高级设置

单击“执行”按钮，即可在管理器窗口的“模型”标签下显示“生成的-选择”模型。右击该模型，在快捷菜单中选择“浏览”命令，即可打开“生成的-选择”窗口，在窗口的“生成”菜单下，选择“过滤器”命令，打开“根据特征选择生成过滤”对话框，选择“包括”模式，并选择“所有标记的字段”单选按钮，勾选“重要”复选框，如图 7.16 所示。然后单击“确定”按钮，即可在数据流区域生成“(已生成)”节点。

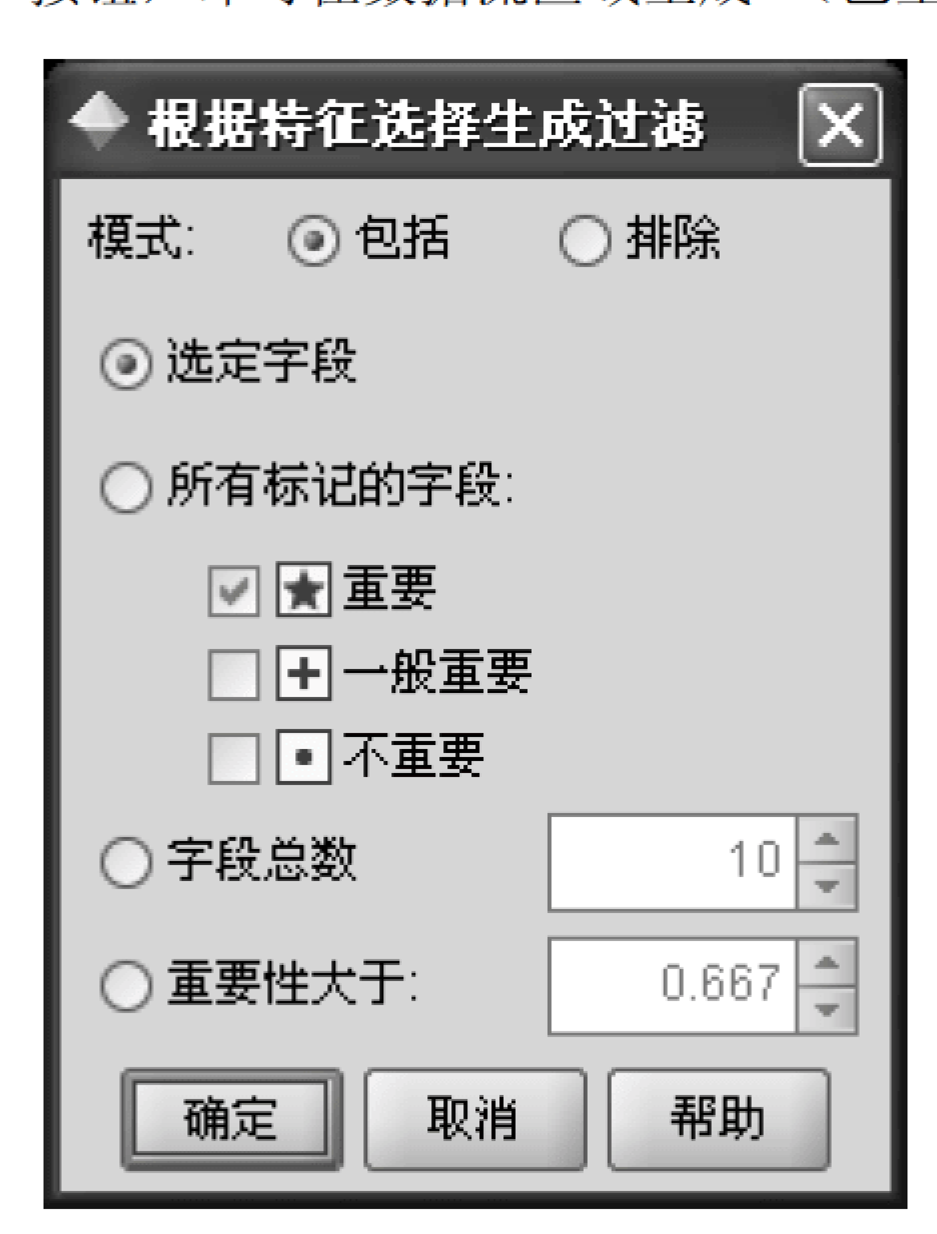


图 7.16　设置字段筛选条件

建立从“类型”节点到“(已生成)”节点之间的连接，用以对字段进行筛选（双击“(已生成)”节点，在弹出的窗口中可以看到，20 个相对不太重要的字段已被过滤）。

将用于建模的 Logistic 节点添加到数据流区域，并建立从“(已生成)”节点到 Logistic 节点的连接，可以看到，Logistic 节点被自动命名为 churn 节点。双击 churn 节点，打开该节点的设置窗口，在该窗口的“模型”标签下，将“过程”设置为“二项式”，如图 7.17 所示。对于二项式过程的方法，有 3 种可选的方法（进入法、前进法、后退法）。

进入法：这是默认的方法，将所有输入字段都作为回归方程中的自变量，构建模型时不进行字段选择。

前进法。采用这种方法，初始模型是最简单的模型（只含有常量），每个步骤会对尚未纳入到模型中的字段进行检验，看它们对模型的改进起多大作用，然后将其中的最佳字段添加到模型中。当无法再添加任何字段或最佳备选字段无法对模型产生足够的改进时，最终模型便已生成。

后退法。采用这种方法时，初始模型包含所有输入字段，只能从模型中删除字段。对模型贡献较小的字段将被逐一删除，直到无法再删除任何字段而不对模型功能造成重大损害，从而生成最终模型。

这里选择“前进法”来构建模型。最后，勾选“将常量纳入方程式”复选框。如图 7.17 所示。



图 7.17 “模型”标签下的设置

切换到“专家”标签下，在“模式”的选项中选择“专家”单选按钮，即可对建模进行高级设置，如图 7.18 所示。



图 7.18 建模的高级设置

单击"输出…"按钮，弹出"Logistic 回归：高级输出"对话框，如图 7.19 所示。



图 7.19 "Logistic 回归：高级输出"对话框

选择"最后步骤"单选按钮，即输出最后的结果，而不必输出中间过程。勾选"参数估计"复选框，其他选项依实际需要灵活选择。其中，"Hosmer-Lemeshow 拟合优度"是一种当自变量较多且多是定距型变量时常用的拟合优度测度方法；"exp(B)的 CI(%)"复选框用来设置是否输出发生比的置信区间。另外，Logistic 回归中还可以进行残差分析，这里不再赘述。

以上设置结束后，单击"确定"按钮。最后，单击"执行"按钮，即可在管理器窗口的"模型"标签下显示生成的 Logistic 回归模型 churn。

2．浏览模型

右击管理器窗口的"模型"标签下的 churn 模型，选择"浏览"命令，可以打开模型的浏览窗口，在"汇总"标签下，显示了本次建模的信息概要，在"高级"标签下，则显示了更加详细的过程和结果。首先显示了"案例处理摘要"，列出关于样本的信息。然后显示了因变量编码以及分类变量（离散型自变量）编码（转换为虚拟变量）。最后给出了在建模开始和建模结束时的模型，即块 0 和块 1。

模型系数的综合检验（即模型线性关系的显著性检验）结果如图 7.20 所示。

		卡方	df	显著性
步骤 9	步骤	3.895	1	.048
	块	264.261	8	.000
	模型	264.261	7	.000

图 7.20 模型系数的综合检验

可以看出，线性关系显著性检验的卡方统计量为 264.261，对应的 p 值接近于 0，小于显著性水平 0.05，所以模型的线性关系显著。

拟合优度的检验结果如图 7.21 所示。

步骤	-2 对数似然值	Cox & Snell R 方	Nagelkerke R 方
9	910.133(a)	.232	.336
a. 因为参数估计的更改范围小于 .000，所以估计在迭代次数 6 处终止。			

图 7.21　模型摘要

可以看出，–2 对数似然值为 910.133，数值较大，相应的似然值则较小，说明模型的拟合优度一般。从 Cox & Snell R^2 和 Nagelkerke R^2 的结果也印证了这一点（Nagelkerke R^2 的值越接近于 1，说明方程的拟合优度越高；越接近于 0，说明方程的拟合优度越低）。

采用模型对训练样本进行预测所得的预测分类表如图 7.22 所示。

	观察值		预测值		
			churn		百分比校正
			.00	1.00	
步骤 9	churn	.00	662	64	91.2
		1.00	145	129	47.1
	总百分比				79.1
a. 切割值为 .500					

图 7.22　预测分类表

可以看出，模型的正确预测率为 79.1%。对于本案例而言，更关心的是要准确地预测到那些流失的客户，即 churn 的观测值为 1 且预测值也为 1 的概率，这里这个百分比为 47.1%。也就是说，通过构建出的 Logistic 回归模型，可以成功地预测出 47.1%的可能流失的客户。

参数估计的结果以及估计参数的统计特征如图 7.23 所示。

可以看出，最后生成的回归方程包括常量和 8 个变量。各数据项的含义依次为：

B：回归参数；

S.E.：回归参数的标准误差；

Wald：Wald 检验统计量的观测值；

df：自由度；

显著性：Wald 检验统计量的概率 p 值；

Exp(B)：发生比。

		B	S.E.	Wald	df	显著性	Exp(B)
步骤 9(a)	tenure	-.037	.005	50.858	1	.000	.964
	employ	-.046	.015	10.023	1	.002	.955
	equip(1)	-.761	.199	14.637	1	.000	.467
	callcard(1)	.947	.248	14.601	1	.000	2.579
	cardmon	.017	.009	3.803	1	.051	1.017
	voice(1)	-.494	.201	6.062	1	.014	.610
	internet(1)	-.538	.201	7.199	1	.007	.584
	lninc	.294	.150	3.841	1	.050	1.341
	常量	-.112	.592	.036	1	.850	.894

a. 在步骤 8 中输入的变量: lninc.

图 7.23　方程中的变量

根据这些信息可以写出回归方程为：

$$\ln\left(\frac{p}{1-p}\right) = -0.112 - 0.037 \times \text{tenure} - 0.046 \times \text{employ} - 0.761 \times \text{equip}(1) + 0.947 \times \text{callcard}(1)$$
$$+0.017 \times \text{cardmon} - 0.494 \times \text{voice}(1) - 0.538 \times \text{internet}(1) + 0.294 \times \text{lninc}$$

这个方程即可用于根据样本的自变量取值来预测因变量的值。例如，将生成的 churn 模型拖入数据流区域，并建立从“：生成的”节点到该节点的连接，然后在其后添加“表”节点，执行该“表”节点，即可生成预测结果，如图 7.24 所示。

表（26 个字段，1,000 条记录） #1

文件　编辑　生成

	ebill	loglong	lninc	custcat	churn	$L-churn	$LP-churn	$LP-0	$LP-1
1	0	1.308	4.159	1	1	0	0.779	0.779	0.221
2	0	1.482	4.913	4	1	0	0.585	0.585	0.415
3	0	2.899	4.754	3	0	0	0.978	0.978	0.022
4	0	2.246	3.497	1	1	0	0.757	0.757	0.243
5	0	1.841	3.401	3	0	0	0.709	0.709	0.291
6	0	2.468	4.357	3	0	0	0.933	0.933	0.067
7	1	2.389	2.944	2	1	0	0.899	0.899	0.101
8	1	1.800	4.331	4	0	0	0.573	0.573	0.427
9	0	2.277	5.112	3	0	0	0.964	0.964	0.036
10	0	3.184	4.277	2	0	0	0.980	0.980	0.020
11	1	1.579	4.828	1	1	1	0.793	0.207	0.793
12	0	1.960	4.382	3	0	0	0.661	0.661	0.339
13	0	2.146	3.611	1	0	0	0.885	0.885	0.115
14	1	2.747	4.745	4	1	0	0.715	0.715	0.285
15	0	1.482	3.219	1	0	0	0.671	0.671	0.329
16	0	1.629	4.317	2	0	0	0.563	0.563	0.437

表　注解

确定

图 7.24　基于 Logistic 回归模型的预测结果

可以看到，预测结果在原有的数据基础上，增加了 4 列信息。其中，$L-churn 是对因变量预测结果（0 或者 1），$LP-0 和$LP-1 分别是因变量的值为“0”和“1”的概率（给定样本的各自变量的值）。$LP-churn 是$LP-0 和$LP-1 中的大者。

第 8 章　神 经 网 络

8.1　神经网络原理

人工神经网络（简称神经网络）是集脑科学、神经心理学和信息科学等多学科的交叉研究领域，是近年来高科技领域的一个研究热点。它的研究目标是通过研究人脑的组成机理和思维方式，探索人类智能的奥秘，进而通过模拟人脑的结构和工作模式，使机器具有类似人类的智能。它已在模式识别、机器学习、专家系统等多个方面得到应用，成为人工智能研究中的活跃领域。

8.1.1　神经网络基本概念

神经网络是一种数学模型，它试图模拟人类大脑的功能。它由大量的处理单元（人工神经元）通过适当的方式互连构成，是一个非线性的自适应系统，用于智能决策或推断。

图 8.1 显示了一个用于预测银行客户信用度的神经网络。

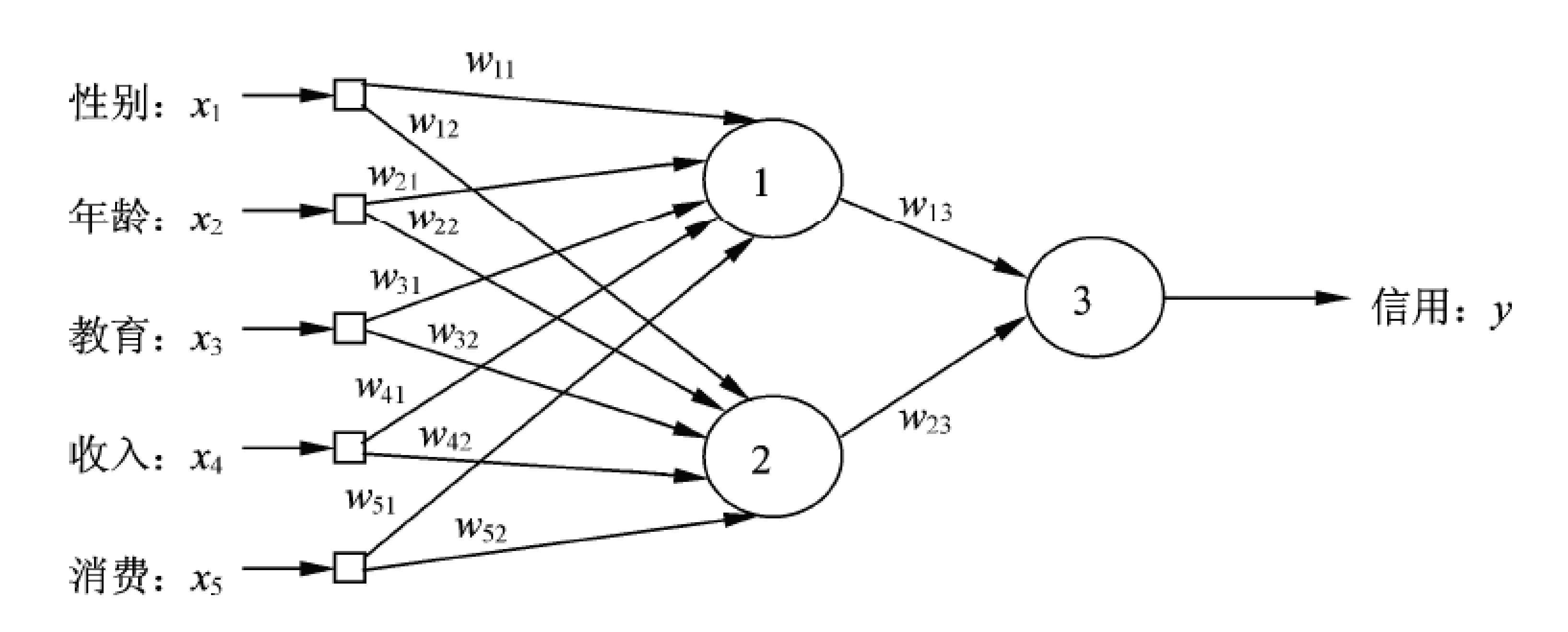


图 8.1　一个神经网络的例子

图中，x_1，x_2，…，x_5 称为网络的输入信号，w_{11}，w_{12}，…，w_{52}，w_{13}，w_{23} 称为突触权值，用来表示各个输入的不同的重要程度。节点 1、节点 2 和节点 3 为神经元。

一个神经网络分为若干层。所有的输入信号构成了输入层（Input Layer），这是神经网络的第一层；神经网络的最后一层是输出层（Output Layer），输出层包含了一个或者多个神经元，如图 8.1 中的节点 3；在输入层和输出层之间的层，称为隐藏层（Hidden Layers），一个神经网络可以包含 1 个或者多个隐藏层，图 8.1 中的节点 1 和节点 2 构成了一个隐藏层。

神经网络的训练过程是，首先构造出网络的基本结构，并初步设定各突触权值。然后将训练样本带入网络中，计算神经网络的输出值，比较网络输出值与期望输出值之间

的误差，然后对各突触权值进行调整，使得网络输出值与期望输出值之间的误差最小。

显然，一个如图 8.1 所示的神经网络被训练完成之后，这个网络模型即可用于对新客户信用度的预测。这也正是神经网络在数据挖掘中的主要应用方式，即用于对新样本目标字段的预测或者分类。

神经网络的结构有多种不同的类型，但无论何种神经网络，其基本单元都是具有计算能力的神经元。因此，要构造一个神经网络系统，首要任务是构造神经元模型。

一个典型的神经元模型如图 8.2 所示。

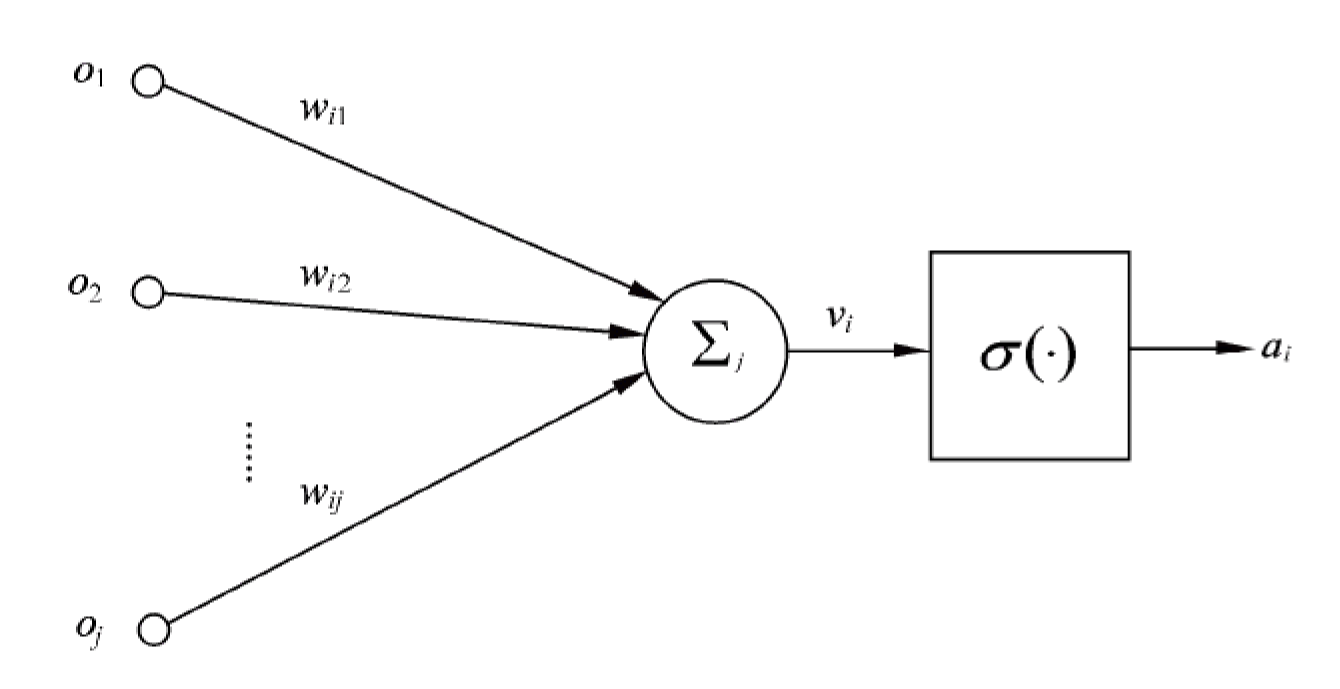


图 8.2　神经元模型

神经元 i 有 j 个输入，分别为 o_1,o_2，$\cdots,o_j$,这 j 个输入来自输入层的节点或者来自前一隐藏层中与神经元 i 相连的 j 个神经元的输出。$w_{i1},w_{i2},\cdots,w_{ij}$ 分别是 j 个输入的突触权值。v_i 是 j 个输入的加权和。$\sigma(\cdot)$ 称为激活函数。a_i 是神经元 i 的输出，所以

$$a_i = \sigma(\sum_j w_{ij} o_j)$$

神经元的输出通常在闭区间[0,1]中，有时也采用另一种区间[–1,1]。

激活函数 $\sigma(\cdot)$ 可以取不同的函数。常用的基本激活函数有以下 3 种。

1. 阈值函数

阈值函数也称为阶跃函数，可以写为：

$$\sigma(v) = \begin{cases} 1 & v \geqslant 0 \\ 0 & v < 0 \end{cases}$$

函数如图 8.3 所示。

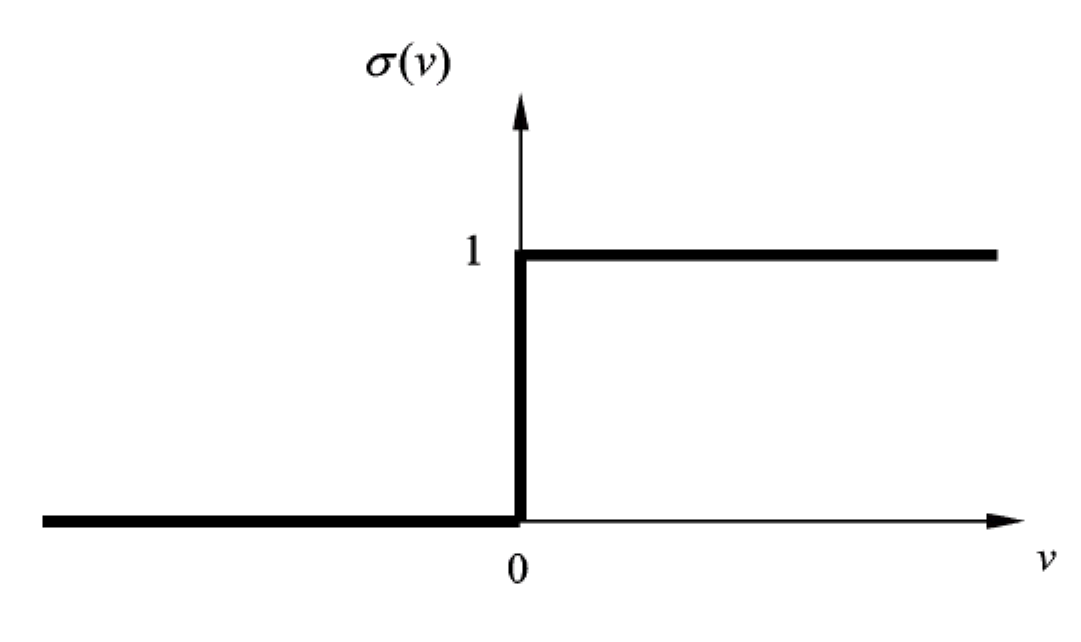


图 8.3　阈值函数

神经元的输出取 0 或者 1，表示神经元兴奋或者抑制。

2．分段线性函数

分段线性函数的表达式为：

$$\sigma(v)=\begin{cases}1 & v\geqslant 0.5\\ v & -0.5<v<0.5\\ 0 & v\leqslant -0.5\end{cases}$$

函数如图 8.4 所示。

3．sigmoid 函数

sigmoid 函数也称为 s 型函数，是构造人工神经网络最常用的激活函数。sigmoid 函数的表达式为：

$$\sigma(v)=\frac{1}{1+\exp(-av)}$$

其中，a 是 sigmoid 函数的斜率参数，通过改变参数 a，可以改变函数的倾斜程度，从而得到不同斜率的 sigmoid 函数，如图 8.5 所示。

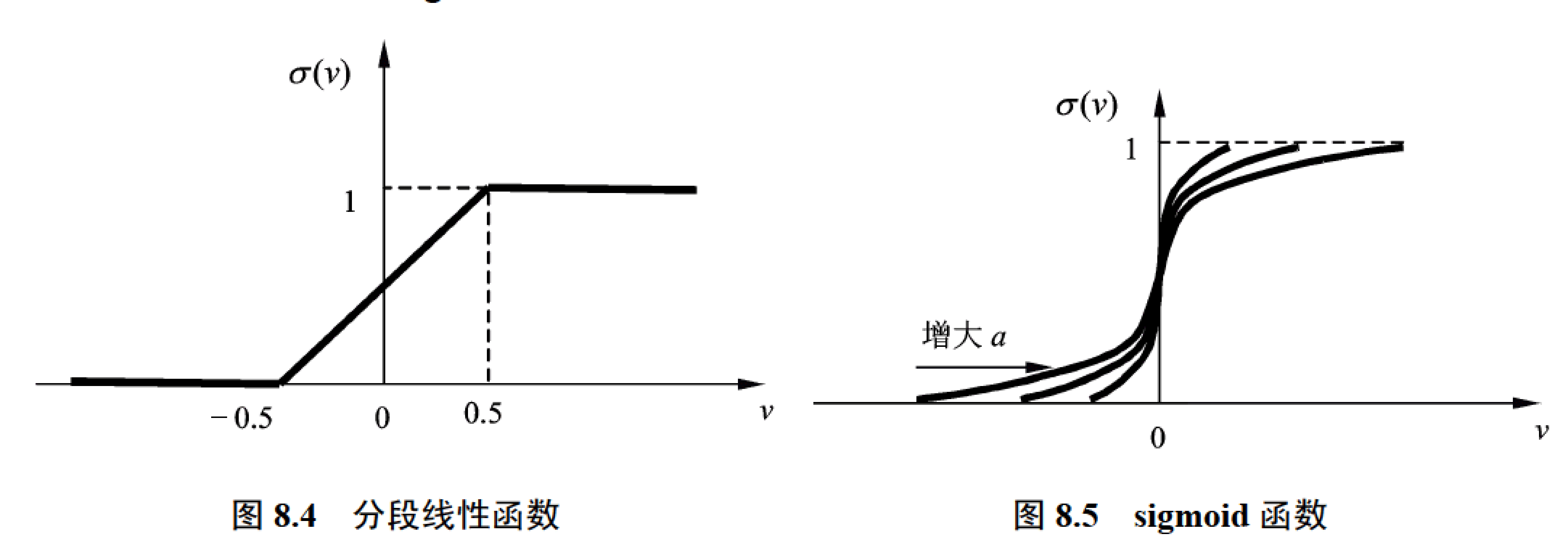


图 8.4　分段线性函数　　图 8.5　sigmoid 函数

8.1.2　神经网络及其学习

将许多神经元按照某种方式连接起来，就成为了神经网络。神经网络的连接方式有很多种，也就形成了不同类型的网络结构。同时，不同类型的神经网络，其相应的训练算法（学习算法）也有较大区别。

1．神经网络的分类

对神经网络的分类可以从多个角度来进行：

（1）从网络性能角度，可以分为连续型和离散型网络、确定性和随机性网络。

（2）从网络结构角度，可以分为前向网络与反馈网络。

（3）从学习方式角度，可以分为有导师学习网络和无导师学习网络。

（4）从连接突触的性质角度，可以分为一阶线性关联网络和高阶非线性关联网络。

就神经网络在数据挖掘中的应用来看，前向神经网络是数据挖掘中应用最广的一种网络，其原理或算法也是很多神经网络模型的基础。后文中详细介绍的多层感知器和径向基函数网络都属于前向神经网络。

前向神经网络又分为单层前向网络和多层前向网络。

所谓单层前向网络，是指网络中只有一层神经元。也就是说，网络包含两层，即输入层和输出层。其中输入层只有输入信号，没有神经元。输出层则由神经元组成。

多层前向网络与单层前向网络的区别在于，多层前向网络的输入层和输出层之间包含一个或多个隐藏层，图 8.1 显示的就是一个 3 层的前向网络，含有一个隐藏层。其中，隐藏层中的神经元被称为隐藏神经元。

在多层前向网络中，输入层的输入信号作为第二层（第一隐藏层）中神经元的输入，第二层神经元的输出再作为第三层神经元的输入，以此类推。因此，信号从输入层输入，经隐藏层传给输出层，最后由输出层得到输出信号。通过加入一个或多个隐藏层，使网络能够提取出更高序的统计，尤其当输入层规模庞大时，隐藏神经元提取高序统计数据的能力显得更加重要。

2．学习算法

神经网络的学习也称为训练，是通过神经网络所在环境的刺激作用调整神经网络的自由参数，使神经网络以一种新的方式对外部环境做出反应的过程。

构造神经网络模型，归根结底是要确定网络中的各突触权值以及相关的自由参数，确定这些数值的方法，就是神经网络的学习算法。不同的学习算法对神经元的突触权值调整的表达式有所不同，也没有任何一种学习算法可以用于所有的神经网络。

从学习的方式上来看，学习可以分为有导师学习和无导师学习两种。

有导师学习又称为有监督学习，在学习时需要给出期望输出（导师信号）。期望输出代表了神经网络执行情况的最佳结果，根据给定的网络输入来调整网络参数，使得网络输出逼近期望输出。

无导师学习包括强化学习和无监督学习（或称为自组织学习）。在强化学习中，对输入输出映射的学习是通过与外界环境的连续作用最小化性能的标量索引而完成的。在无监督学习中没有外部评价来监督学习的过程，而是提供一个关于网络学习表示方法质量的测量尺度，根据该尺度将网络的自由参数最优化。

如前所述，不同类型的神经网络，其学习算法各不相同。对于前向神经网络，大多采用纠错学习算法，也称为 Delta 规则或者 Widrow-Hoff 规则，该算法的基本思想描述如下：

对于神经元 j，第 n 次训练其产生的输出为 $y_j(n)$，这是神经网络根据输入通过计算得到的结果，是网络的输出。与这个输入相对应的期望输出即为 $d(n)$，那么网络输出和期望输出之间的误差为 $e(n)=d(n)-y_j(n)$。现在，学习的任务就是要调整突触权值，使得误差 $e(n)$减小。为此，可设定代价函数 $E(n)$

$$E(n)=\frac{1}{2}e^2(n)$$

反复调整突触权值，使代价函数达到最小，或者突触权值稳定下来，就完成了学习任务。

用 w_{ji} 表示神经元 i 到神经元 j 的突触权值，在第 n 次训练时，这个突触权值要进行调整，其调整量为：$\Delta w_{ji}(n)=\eta e(n)y_i(n)$，其中，$\eta$ 称为学习率参数，$y_i(n)$ 表示此时神经元 i 的输出（也就是神经元 j 的输入）。这表明，对神经元突触权值的调整与突触误差信号和输入信号成比例。在得到 $\Delta w_{ji}(n)$ 之后，定义突触权值的校正值为：

$$w_{ji}(n+1)=w_{ji}(n)+\Delta w_{ji}(n)$$

其中，$w_{ji}(n)$ 和 $w_{ji}(n+1)$ 分别是突触权值的旧值和新值。

8.2 多层感知器与 RBF 网络

多层感知器与 RBF 神经网络是目前应用最为广泛的两种前向型神经网络。

8.2.1 多层感知器

1. 多层感知器简介

多层感知器是一种典型的多层前向网络，在有监督学习的方式下，通过“误差反向传播算法”来训练多层感知器。

多层感知器由输入层、隐藏层和输出层组成，其中隐藏层可以是一层或者多层。输入层的节点数为输入信号的维数，隐藏层个数以及每个隐藏层包含隐藏神经元的个数视具体情况而定，输出层神经元的个数为输出信号的维数。

多层感知器有 3 个突出的特点：

（1）多层感知器包含有一层或者多层隐藏层，隐藏神经元可以从输入信号中提取更多有用的信息，使网络可以完成更加复杂的行为。

（2）多层感知器中每个神经元所使用的激活函数是可微的 sigmoid 函数：

$$\sigma(x)=\frac{1}{1+e^{-x}}$$

这里，x 是神经元全部输入的加权和，$\sigma(x)$ 则是神经元的输出。

（3）多层感知器具有高度的连通性，这是由突触决定的。网络连接的改变需要突触连接数量及其权值的改变。

正是由于上述特点，使得多层感知器具有强大的计算能力，从而能够通过训练从经验中进行学习。

为了更加深入地理解多层感知器，还必须理解在多层感知器中流动着的两种信号以及它们完全相反的流动方向，如图 8.6 所示。这两种信号分别是“函数信号”和“误差信号”。

函数信号是从网络的输入层而来的信号，通过网络中神经元之间的传播，到达网络输出层称为输出信号，它是正向传播的。函数信号在向前传播的过程中，网络中的突触

权值是固定不变的。

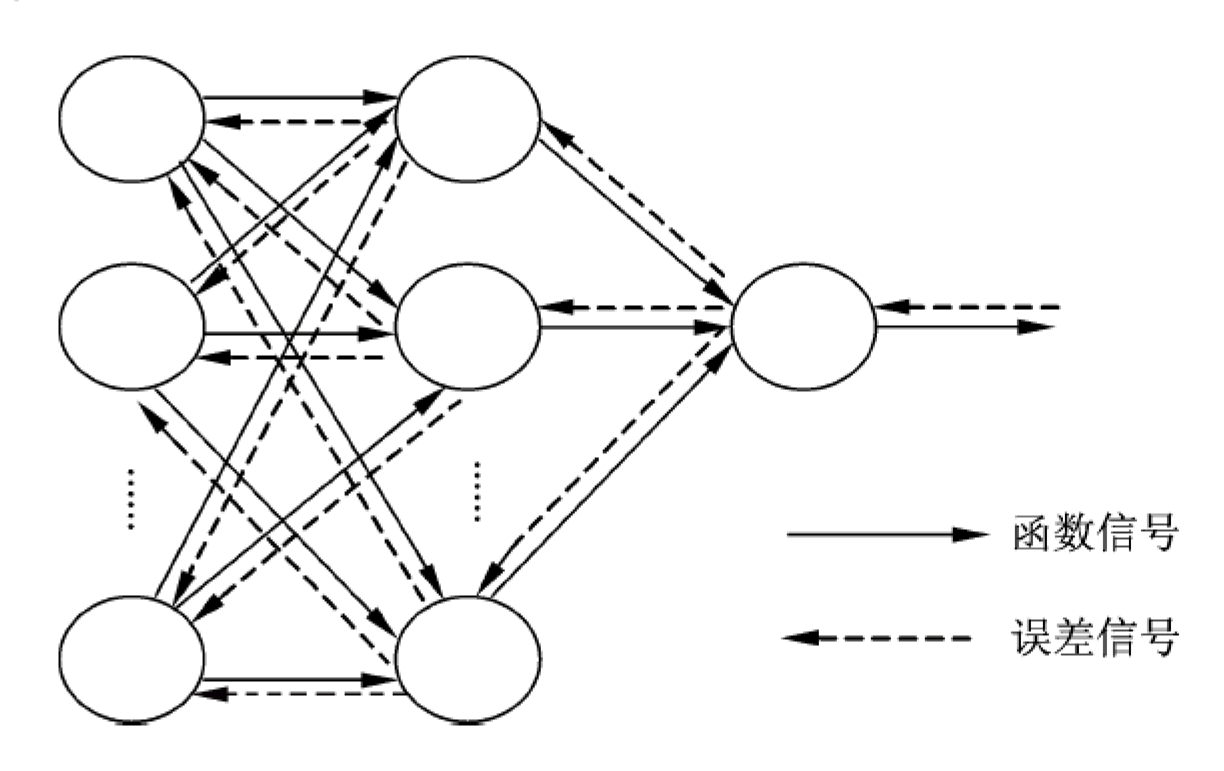


图 8.6　函数信号与误差信号

误差信号是从网络的输出层而来的信号。网络的输出与期望输出之间的差值即为误差信号，误差信号由输出层逐层反向传播。在误差信号反向传播的过程中，网络中的突触权值要根据误差信号进行调节，通过权值的不断修正，使得网络的输出不断向期望输出逼近。

因此，在多层感知器的每个神经元中，都必须能够完成以下两个计算功能：

（1）计算每个神经元输出的函数信号，它表现为关于该神经元所有输入信号以及相应突触权值的一个连续非线性函数。

（2）计算每个神经元反向传播的误差信号，并进一步根据误差信号计算每个与该神经元相连的突触权值的改变量。这也是每个神经元最主要的计算内容。

2．多层感知器的输入

在开始进行神经网络的训练之前，有必要先了解一下多层感知器对输入信号的要求，并不是任何格式的输入信号都可以直接用于神经网络的训练中。

传递给神经元的输入信号的值必须是数值型的且落在[0,1]闭区间中。因此，对于不满足这一要求的连续型数据和离散型数据，首先要进行数据的转换。

（1）连续型数据的转换。

可以通过下面这个公式进行转换：

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

其中，$x_{\min}$ 和 $x_{\max}$ 分别是属性 X 的全部取值中的最小值和最大值，x_i 是某个样本的属性 X 的取值，x_i' 就是转换后的属性值。显然，x_i' 落在[0,1]闭区间中。这一转换的另一个好处是使得所有连续型属性具有了统一的测度。

（2）离散型数据的数字化。

数据库中有些属性是离散型的，例如“颜色”的取值属于一个域{黑，白，紫，红}。

① 如果域中的取值比较少。

在 Clementine 中，用一个数值来表示离散型域中的每一个值。这样原始的离散型字段被转化为若干个数值型字段（派生字段）。对于某个字段的取值，其对应的派生字段取

值为 1，其他派生字段取值为 0。所以这些派生字段也称为指示器字段。

例如，在下面这些数据中，X 是一个离散型字段，值域为$\{A, B, C\}$：

Record#	X	X_1'	X_2'	X_3'
1	B	0	1	0
2	A	1	0	0
3	C	0	0	1

其中，原始属性 X 被编码为 3 个派生属性：X_1'，X_2'，X_3'。X_1'是 A 的指示器，X_2'是 B 的指示器，X_3'是 C 的指示器。

② 如果域中的取值比较多。

对于上面这种方法，每一个取值都需要一个派生属性，所以如果取值较多，将有太多的派生属性，这是不行的。这时可采用二元集编码，即每个取值用一个二进制数来表示，那么 4 个取值只需要 2 个派生属性。例如：

Record#	X	X_1	X_2
1	A	0	1
2	B	1	0
3	C	1	1

③ 标志型字段的编码。

标志型字段只有两个值：true 和 false。用 1 表示 true，用 0 表示 false，用 0.5 来表示缺失值。

3. 训练神经网络

训练开始的时候，网络中的所有权值都被随机地设置为一个属于[–0.5,0.5]的值。

训练的过程包括几轮。在每一轮中，从训练数据集中随机地选取 n 个记录出来（由于是随机选取，所以有的记录会被多次选取，有的记录则从未被选取），把这 n 个记录依次代入网络中。

对于每一个记录，记录中各属性的值被当作输入信号输入给神经网络，然后网络中的神经元根据给自己的输入及相应的突触权值来计算它们各自的输出，依据的计算公式为：

$$a_i = \sigma(\sum_j w_{ij} o_j)$$

这里，a_i 是神经元 i 的输出，w_{ij} 是神经元 j 到神经元 i 的连接的权值，o_j 是神经元 j 的输出。$\sigma(x)$ 是激活函数：$\sigma(x) = \dfrac{1}{1+\mathrm{e}^{-x}}$。

当逐层的神经元计算出各自的输出后，最终在输出层神经元处将得到对当前记录目标字段的预测值。把这个预测值和当前记录的实际值相比较，就产生了预测误差。这个预测误差将被反馈到网络中去更新所有的突触权值。

假设现在将记录 p 代入网络，每个神经元都计算出了各自的输出，下面开始由输出层反向逐层的神经元计算各自的反向传播误差。对于一个神经元 j，它的反向传播误差 δ_{pj} 因其所在的层不同，其计算方法也不同：

（1）如果神经元 j 是输出层的神经元，那么其反向传播误差为：

$$\delta_{pj}=(t_{pj}-o_{pj})\sigma'(a_j)$$

其中，t_{pj} 是记录 p 的目标字段的实际值（即神经元的期望输出），o_{pj} 是神经元 j 对记录 p 的目标字段的计算输出（即神经元的预测输出），$\sigma'(a_j)$ 是激活函数的一阶导数 $\sigma'(x)$ 在 $x=a_j$ 处的值，a_j 是神经元 j 的输入加权和。显然，$o_{pj}=\sigma(a_j)$。

由于 $\sigma'(x)=\left(\frac{1}{1+e^{-x}}\right)'=\left(\frac{1}{1+e^{-x}}\right)\left(1-\frac{1}{1+e^{-x}}\right)=\sigma(x)(1-\sigma(x))$，所以

$$\delta_{pj}=(t_{pj}-o_{pj})\sigma'(a_j)=(t_{pj}-o_{pj})o_{pj}(1-o_{pj})$$

（2）如果神经元在隐藏层，那么其反向传播误差为

$$\delta_{pj}=\sigma'(a_j)\sum_k \delta_{pk}w_{jk}=o_{pj}(1-o_{pj})\sum_k \delta_{pk}w_{jk}$$

其中，k 表示神经元 j 有 k 个反向输入的误差，w_{kj} 是神经元 j 与神经元 k 连接的权值，o_{pj} 是神经元 j 对于记录 p 的计算输出。如图 8.7 所示，隐藏层神经元 j 有 k 个反向传播的输入误差。

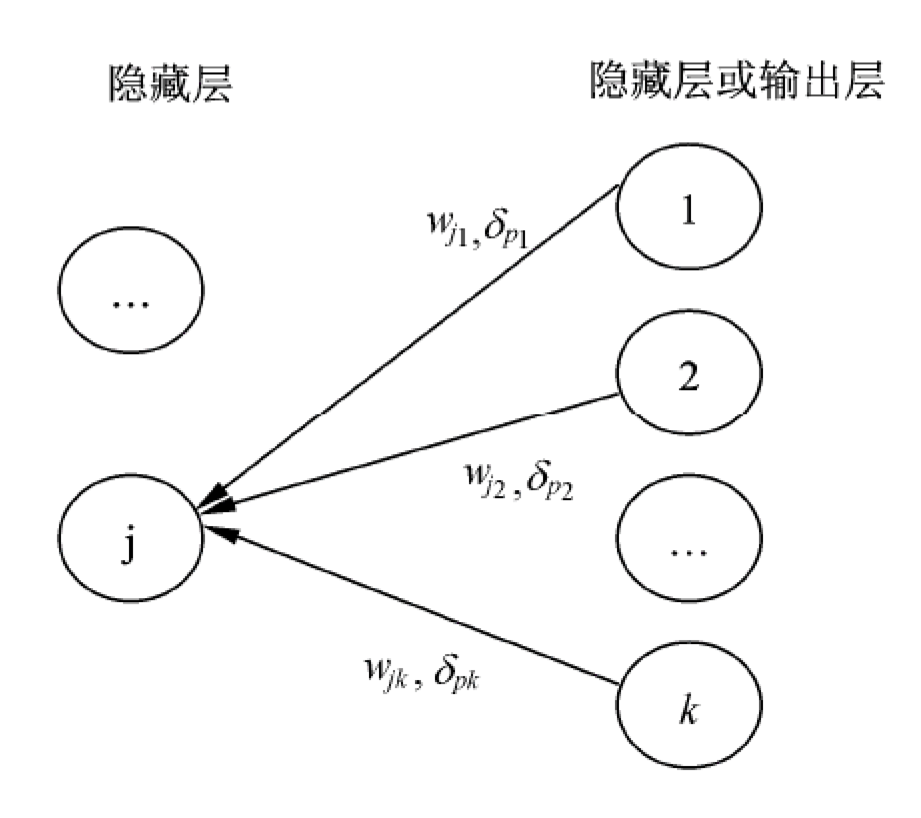


图 8.7　隐藏层神经元的反向传播误差

在反向传播误差计算结束后，即可计算突触权值的改变量 Δw。神经元 i 到神经元 j 的突触权值的改变量为

$$\Delta w_{ij}(n+1)=\eta\delta_{pj}o_{pi}+\alpha\Delta w_{ij}(n)$$

其中：η 是学习率参数，$0<\eta<1$；

δ_{pj} 是神经元 j 的反向传播误差；

o_{pi} 是神经元 i 的输出；

α 称为动量常数，通常是正数，在神经网络训练过程中固定不变；

$\Delta w_{ij}(n)$ 是 w_{ij} 在上一次的改变量（n 表示“某一次”，所以 n+1 代表“之后的一次”）。

在 Clementine 中，学习率参数 η 的值在训练中的每一轮都会改变。η 的改变是这样来实现的：首先由用户自定义 3 个参数，initial eta、low eta、high eta。η 开始于 initial eta，然后以对数方式不断减小到 low eta，然后又还原到 high eta，再又不断减小到 low eta。η

的值这样来计算：

$$\eta(t)=\eta(t-1)\cdot\exp\left(\log\left(\frac{\eta_{\text{low}}}{\eta_{\text{high}}}\right)/d\right)$$

其中，d 是用户指定的 η 衰减周期数，也就指定了 η 降低的速率，该速率以从 η_{high} 到 η_{low} 的周期数表示。当 $\eta(t-1)<\eta_{\text{low}}$ 时，$\eta(t)$就被设置为 η_{high}。循环往复，直到训练结束，如图 8.8 所示。

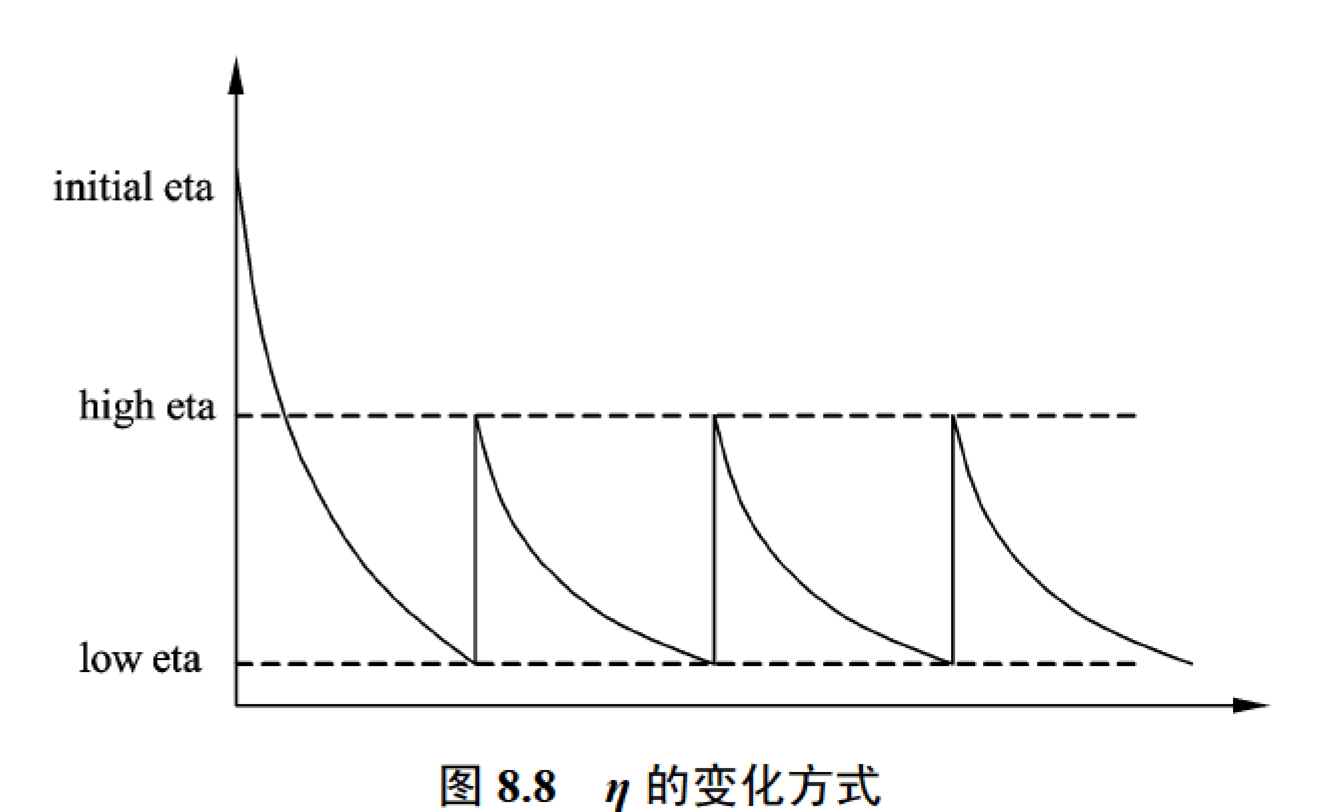


图 8.8　η 的变化方式

一个记录被提交给网络后，通过以上计算，每个权重值被立即更新为

$$w_{ij}(n+1)=w_{ij}(n)+\Delta w_{ij}(n+1)$$

当满足设定的训练停止准则后，训练任务结束，形成最终的神经网络模型。

下面通过一个实例来演示上面的计算过程。

图 8.9 所示的多层感知器网络包含 3 层：输入层、输出层和一个隐藏层。假设当前各突触权值为：w_{1j}=0.2，w_{1i}=0.1，w_{2j} =0.3，w_{2i}=–0.1，w_{3j}=–0.1，w_{3i}=0.2，w_{jk} =0.1，w_{ik} =0.5。

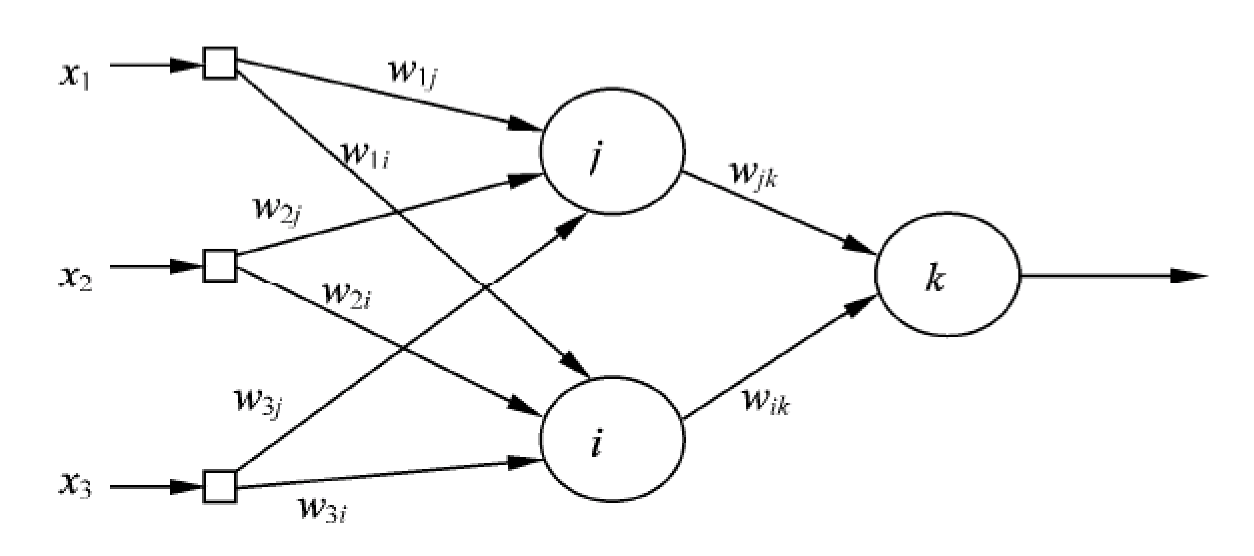


图 8.9　多层感知器实例

此时将输入信号 x_1=1，x_2=0.4，x_3=0.7 代入网络，那么神经元 i 和 j 的输入分别为

$$a_j=1\times0.2+0.4\times0.3+0.7\times(-0.1)=0.25$$

$$a_i=1\times0.1+0.4\times(-0.1)+0.7\times0.2=0.2$$

神经元 i 和 j 的输出分别为

$$o_j=\sigma(a_j)=\frac{1}{1+\mathrm{e}^{-0.25}}=0.562\;;\qquad o_i=\sigma(a_i)=\frac{1}{1+\mathrm{e}^{-0.2}}=0.55$$

神经元 k 的输入为

$$a_k=0.562\times0.1+0.55\times0.5=0.331$$

神经元 k 的输出为

$$o_k = \sigma(a_k) = \frac{1}{1+\mathrm{e}^{-0.331}} = 0.582$$

设神经元 k 的期望输出 t_k=0.65，那么可计算神经元 k 的反向传播误差为

$$\delta_k = (t_k - o_k)o_k(1-o_k) = (0.65-0.582)\times 0.582\times(1-0.582) = 0.017$$

隐藏层的神经元 j 和 i 的反向传播误差分别为

$$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{jk} = 0.562\times(1-0.562)\times(0.017\times 0.1) = 0.000418$$

$$\delta_i = o_i(1-o_i)\sum_k \delta_k w_{ik} = 0.55\times(1-0.55)\times(0.017\times 0.5) = 0.002104$$

设动量常数 α 设定为 0.05，当前的学习率参数 η 为 0.5，上一次的权值改变量分别为

$\Delta w_{1j}(n)$=0.001，$\Delta w_{1i}(n)$=0.003，$\Delta w_{2j}(n)$= –0.005，$\Delta w_{2i}(n)$= 0.01，$\Delta w_{3j}(n)$=0.001，$\Delta w_{3i}(n)$= –0.01，$\Delta w_{jk}(n)$=0.013，$\Delta w_{ik}(n)$= –0.025。

那么当前的权值改变量为

$$\Delta w_{1j}(n+1) = \eta\delta_j o_1 + \alpha\Delta w_{1j}(n) = 0.5\times 0.000418\times 1 + 0.05\times 0.001 = 0.000259$$

$$\Delta w_{1i}(n+1) = \eta\delta_i o_1 + \alpha\Delta w_{1i}(n) = 0.5\times 0.002104\times 1 + 0.05\times 0.003 = 0.001202$$

$$\Delta w_{2j}(n+1) = \eta\delta_j o_2 + \alpha\Delta w_{2j}(n) = 0.5\times 0.000418\times 0.4 + 0.05\times(-0.005) = 0.000166$$

$$\Delta w_{2i}(n+1) = \eta\delta_i o_2 + \alpha\Delta w_{2i}(n) = 0.5\times 0.002104\times 0.4 + 0.05\times 0.01 = 0.000921$$

$$\Delta w_{3j}(n+1) = \eta\delta_j o_3 + \alpha\Delta w_{3j}(n) = 0.5\times 0.000418\times 0.7 + 0.05\times 0.001 = 0.000196$$

$$\Delta w_{3i}(n+1) = \eta\delta_i o_3 + \alpha\Delta w_{3i}(n) = 0.5\times 0.002104\times 0.7 + 0.05\times(-0.01) = 0.000236$$

$$\Delta w_{jk}(n+1) = \eta\delta_k o_j + \alpha\Delta w_{jk}(n) = 0.5\times 0.017\times 0.562 + 0.05\times 0.013 = 0.005427$$

$$\Delta w_{ik}(n+1) = \eta\delta_k o_i + \alpha\Delta w_{ik}(n) = 0.5\times 0.017\times 0.55 + 0.05\times(-0.025) = 0.003425$$

从而可计算出新的突触权值：

w_{1j}=0.2+0.000259=0.200259；w_{1i}=0.1+0.001202=0.101202；

w_{2j} =0.3+0.000166=0.300166；w_{2i}=–0.1+0.000921=–0.099079；

w_{3j}=–0.1+0.000196=–0.099804；w_{3i}=0.2+0.000236=0.200236；

w_{jk} =0.1+0.005427=0.105427；w_{ik} =0.5+0.003425=0.503425。

8.2.2 径向基函数网络

Powell 在 1985 年提出了多变量插值的径向基函数（Radial-Basis Function，RBF）方法，Broomhead 和 Lowe 在 1988 年首先将 RBF 应用于神经网络的设计，构成了径向基函数神经网络，即 RBF 神经网络。

RBF 神经网络是一种 3 层前向网络，由输入层、隐藏层和输出层构成。同层之间没有连接，相邻两层之间则完全连接。输入层由信号源节点组成，节点的数目等效于所研究问题的独立变量数目。第二层为隐藏层，其包含的隐藏神经元的数量由所描述的问题而定，隐藏神经元以基函数为激活函数，通过径向基函数对输入产生非线性映射。输出层神经元对隐藏层的输出进行线性加权组合。

RBF 神经网络具有较强的输入、输出映射功能，并且理论证明在前向网络中 RBF

神经网络是完成映射功能的最优网络。因此，RBF 神经网络以其简单的结构、快速的训练过程和良好的推广能力等诸多优点在许多应用领域取得了巨大的成功，特别是在模式分类和函数逼近方面。

根据隐藏层所包含的神经元数量不同，RBF 神经网络可以分为正规化网络（Regularization Network）和广义网络（Generalized Network）两种。在正规化网络中，隐藏层的神经元数量与训练样本的个数相同，因此适用于小样本数据集。当样本量较大时，则应该采用广义网络。由于在数据挖掘中往往处理的是海量样本，因此在实际应用中更多地会选用广义径向基函数网络。

如图 8.10 所示，广义径向基函数网络的输入层由 N 个输入信号组成。隐藏层由 J（$J<N$）个神经元组成，任一隐藏神经元用 j 表示，第 j 个隐藏神经元的激励输出为“基函数”$\phi(r,c_j)$，其中 r 是神经元的输入向量（例如一个训练样本$[r_1,r_2,\cdots,r_N]$），$c_j=[c_{j1}, c_{j2},\cdots, c_{jN},]$（$j=1$，2，…，$J$）是基函数的中心。基函数一般选择格林函数中的高斯函数：

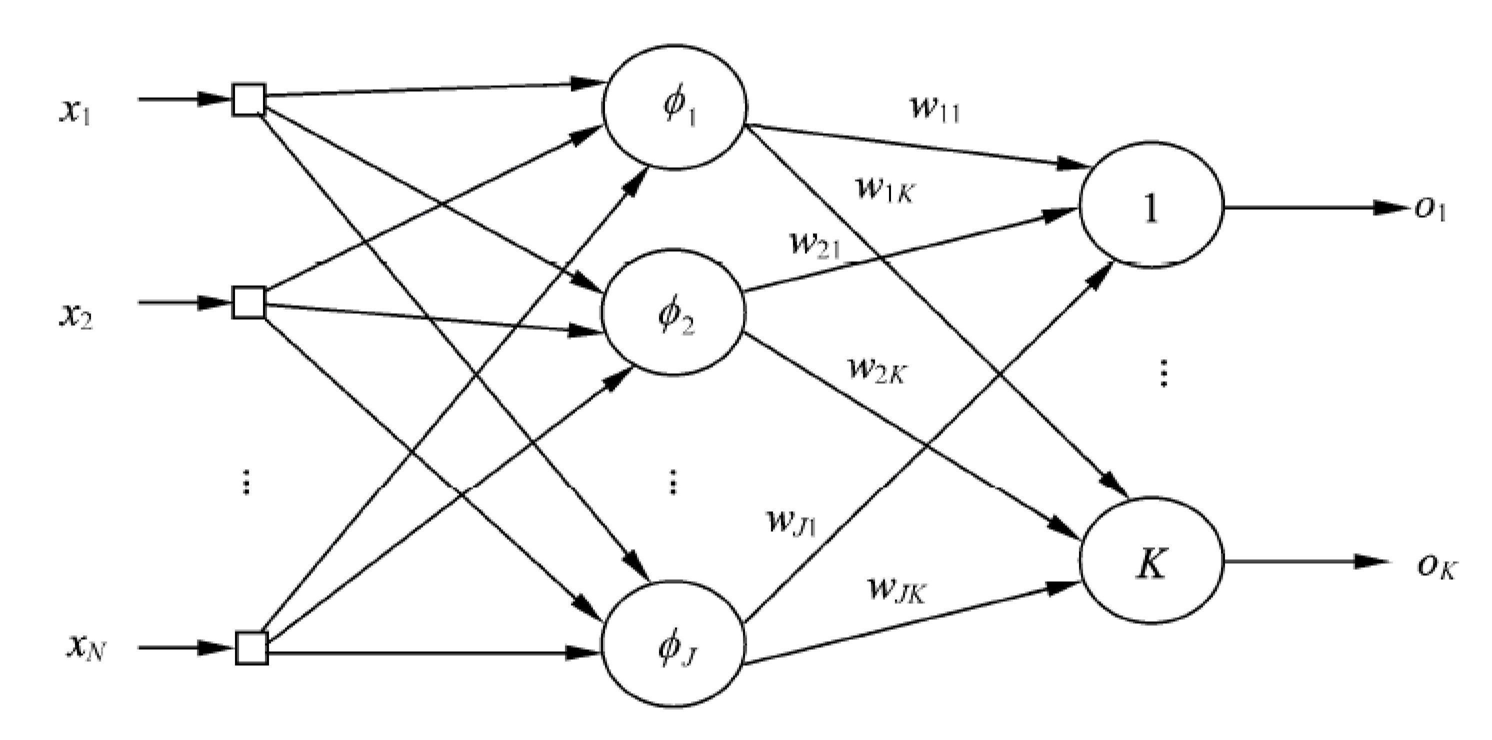


图 8.10　广义径向基函数网络

$$\phi(r)=\exp\left(-\frac{(r-c)^2}{2\sigma^2}\right)$$

其中，c 是高斯函数的中心，σ 为方差。其函数图像如图 8.11 所示。

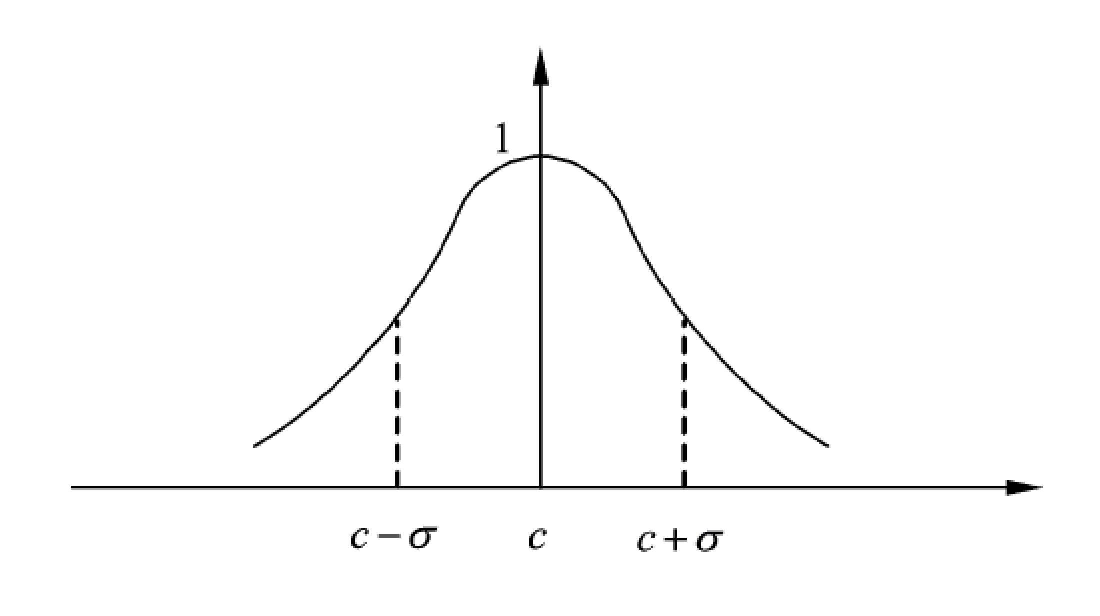


图 8.11　高斯函数

因此，在应用高斯函数时，必须首先确定高斯函数的中心 c 和方差 σ。

确定高斯函数中心的方法主要有 4 种：随机选取中心法、自组织选取中心法、有监督选取中心法和正交最小二乘法。在 Clementine 中采用的是自组织选取中心法，是用 K-Means 聚类算法来实现的。也就是说，首先根据训练样本集用 K-Means 聚类算法生成

J 个聚类，每个聚类对应着隐藏层的一个神经元，并且这个聚类的质心向量作为所对应神经元的高斯函数的中心。K-Means 聚类算法请参见第 4.2 节。在生成这些聚类时，Clementine 设定的停止准则为：①最大迭代次数为 10；②差异容忍度为 0.000001。当满足任一准则时，停止迭代。

高斯函数的方差 σ 决定了基函数的作用范围。σ 值由小变大，高斯函数的响应面则从陡峭向平坦变化，响应面越陡峭，神经网络对样本与高斯函数中心的区分越灵敏。但是 σ 过小，神经网络对噪声太敏感，易失真，而 σ 过大，则会使得神经网络丧失区分和拟合的能力，因此 RBF 网络需要选择合适的 σ 值。

针对所研究问题的实际情况，方差 σ 值可用不同的方法来确定和调整。例如可以采用：

$$\sigma_1 = \sigma_2 = \cdots = \sigma_J = \frac{d_{\max}}{\sqrt{2J}}$$

其中，$d_{\max}$ 是所选取中心之间的最大距离（即距离最远的两个聚类质心之间的距离），J 是聚类数（即隐藏层神经元数目）。

在 Clementine 中，σ 值是由如下公式来确定的：

$$\sigma_1 = \sigma_2 = \cdots = \sigma_J = \sqrt{\frac{d_1 + d_2}{2}}$$

其中，d_1 和 d_2 分别是聚类质心之间的最大距离和第二大的距离。

现在，高斯函数的中心 c 和方差 σ 确定之后，可以得到任一隐藏神经元 j 的激励响应为：

$$a_j = \exp\left(-\frac{\| r - c \|^2}{2\sigma_j^2}\right)$$

其中，r 是神经元 j 的输入向量，c 是该神经元的质心向量。$\| r - c \|$ 实际上是 r 与 c 之间的欧氏距离（Euclid Distance）。

在 Clementine 中，对上式稍作调整，增加了一个 RBF 重叠系数 h，神经元 j 的激励响应计算公式变为：

$$a_j = \exp\left(-\frac{\| r - c \|^2}{2\sigma_j^2 h}\right)$$

这里的 RBF 重叠系数 h 是一个正实数值，用于控制聚类之间的重叠数量。通过增加这个系数，可以增加每个隐藏神经元相关联的聚类的大小，从而使输入的每个训练样本能够影响到距离较远的聚类。

在 RBF 网络中，从隐藏层神经元 j 到输出层神经元 k 之间的权值记为 w_{jk}。输出层神经元对隐藏层的输出进行线性加权组合。因此，一个输出层神经元 k 的输出为：

$$o_k = \sum_j w_{jk} a_j$$

神经网络的训练，最终要通过训练来确定最优的突触权值。RBF 网络中突触权值的训练同样采用反向误差传播方法，相当于训练一个二层前向网络。在训练之初，先将所有的权值初始化为 $-0.001 \leqslant w_{jk} \leqslant 0.001$ 中的值，然后在每次输入训练样本后的训练周期

中依下式进行更新：

$$w_{jk}(t) = w_{jk}(t-1) + \Delta w_{jk}(t)$$

其中，改变率 $\Delta w_{jk}(t)$ 依下式计算：

$$\Delta w_{ij}(t) = \eta(r_k - o_k)a_j + \alpha\Delta w_{jk}(t-1)$$

与多层感知器一样，这里的 η 是学习率参数，α 为动量常数。但与多层感知器的训练不同的是，这里的 η 在训练过程中固定不变。

以上介绍的是最常用的 RBF 网络训练方法。在实际应用中，基函数除了选用高斯函数，也常使用多二次函数和逆多二次函数，它们都是对中心点径向对称的函数：

多二次函数：

$$\phi(r) = \sqrt{r^2 + c^2}$$

逆多二次函数：

$$\phi(r) = \frac{1}{\sqrt{r^2 + c^2}}$$

另外，在上面介绍的方法中，基函数的方差都是相同的。其实每个基函数都有自己的方差，在训练过程中也可以根据自身的情况来分别设定。

最后来总结一下 RBF 网络与多层感知器之间的区别。虽然 RBF 网络与多层感知器都是非线性的层状前向网络，但在具体实现上存在下列几个显著的区别：

（1）RBF 网络只有一个隐藏层，而多层感知器则可以有多个隐藏层。

（2）在多层感知器中，所有神经元（隐藏层神经元和输出层神经元）的计算模型都是相同的，而在 RBF 网络中，隐藏层神经元和输出层神经元的计算模型不同，作用也不同。

（3）从神经元的计算模型可以看出，RBF 网络中隐藏层神经元的计算是非线性的，而输出层的计算则是线性的。对于多层感知器而言，当用于解决模式分类问题时，它的隐藏层和输出层通常选为非线性的，当用于解决非线性回归问题时，其输出层通常选择线性的。

（4）RBF 网络隐藏层神经元的基函数计算的是输入向量和质心的欧氏距离，而多层感知器神经元的激励函数计算的是输入信号和连接权值间的内积。

（5）多层感知器是对非线性映射的全局逼近，而 RBF 网络使用局部指数衰减的非线性函数（如高斯函数）对非线性输入-输出映射进行局部逼近。这意味着当逼近一个非线性的输入-输出映射时，在相同精度要求下，多层感知器需要的参数要比 RBF 网络所需要的参数数量少。

由于 RBF 网络能够逼近任意的非线性函数，可以处理系统内在的难以解析的规律性，并且具有极快的学习收敛速度，因此 RBF 网络有较为广泛的应用。

8.2.3 在 Clementine 中应用神经网络

在 Clementine 中，构建神经网络模型是由“神经网络”节点来完成的。在利用该节点训练神经网络时，对输入字段的类型没有限制，可以处理数字、符号或标志型输入和输出。在训练之前，必须为神经网络节点指定一个或多个方向为“输入”的字段，以

及一个或多个方向为“输出”的字段。设置为“双向”或“无”的字段将被忽略。

本小节描述一个根据数据样本集来训练神经网络的案例，在本例中，将根据全国各省的多项经济指标分别构建一个多层感知器和一个 RBF 网络，并将其用于对 GDP 总量的预测分析。样本数据集存放在“经济发展基本信息.xls”文件中，如表 8-1 所示。

表 8-1 经济发展基本信息样本

地区	居民消费水平	固定资产投资	职工平均工资	货物周转量	居民消费价格指数	商品零售价格指数	工业总产值	GDP
北京	2505	519.01	8144	373.9	117.3	112.6	843.43	1394.89
天津	2720	345.46	6501	342.8	115.3	110.6	582.51	920.11
河北	1258	704.87	4839	2033.3	115.2	115.8	1234.85	2849.52
山西	1250	290.9	4721	717.3	116.9	115.6	697.25	1092.48
内蒙古	1387	250.23	4134	781.7	117.5	116.8	419.39	832.88
辽宁	2397	887.99	4911	1371.1	116.1	114	1840.55	2793.37
吉林	1872	320.45	4430	497.4	115.2	114.2	762.47	1129.2
黑龙江	2334	435.73	4145	824.8	116.1	114.3	1240.37	2014.53
上海	5343	996.48	9279	207.4	118.7	113	1642.95	2462.57
江苏	1926	1434.95	5943	1025.5	115.8	114.3	2026.64	5155.25
浙江	2249	1006.39	6619	754.4	116.6	113.5	916.59	3524.79
安徽	1254	474	4609	908.3	114.8	112.7	824.14	2003.58
福建	2320	553.97	5857	609.3	115.2	114.4	433.67	2160.52
江西	1182	282.84	4211	411.7	116.9	115.9	571.84	1205.11
山东	1527	1229.55	5145	1196.6	117.6	114.2	2207.69	5002.34
河南	1034	670.35	4344	1574.4	116.5	114.9	1367.92	3002.74
湖北	1527	571.68	4685	849	120	116.6	1220.72	2391.42
湖南	1408	422.61	4797	1011.8	119	115.5	843.83	2195.7
广东	2699	1639.83	8250	656.5	114	111.6	1396.35	5381.72
广西	1314	382.59	5105	556	118.4	116.4	554.97	1606.15
海南	1814	198.35	5340	232.1	113.5	111.3	64.33	364.17
四川	1261	822.54	4645	902.3	118.5	117	1431.81	3534
贵州	942	150.84	4475	301.1	121.4	117.2	324.72	630.07
云南	1261	334	5149	310.4	121.3	118.1	716.65	1206.68
西藏	1110	17.87	7382	4.2	117.3	114.9	5.57	55.98
陕西	1208	300.27	4396	500.9	119	117	600.98	1000.03
甘肃	1007	114.81	5493	507	119.8	116.5	468.79	553.35
青海	1445	47.76	5753	61.6	118	116.3	105.8	165.31
宁夏	1355	61.98	5079	121.8	117.1	115.3	114.4	169.75
新疆	1469	376.95	5348	339	119.7	116.7	428.76	834.57

在构建神经网络模型时，设“居民消费水平”为 x_1、“固定资产投资”为 x_2、“职工平均工资”为 x_3、“货物周转量”为 x_4、“居民消费价格指数”为 x_5、“商品零售价格指

数”为 x_6、“工业总产值”为 x_7，作为神经网络的输入，即输入层节点数为 7。设 GDP 为 y 作为网络的输出，即输出层只有一个神经元。

完整的数据流如图 8.12 所示。

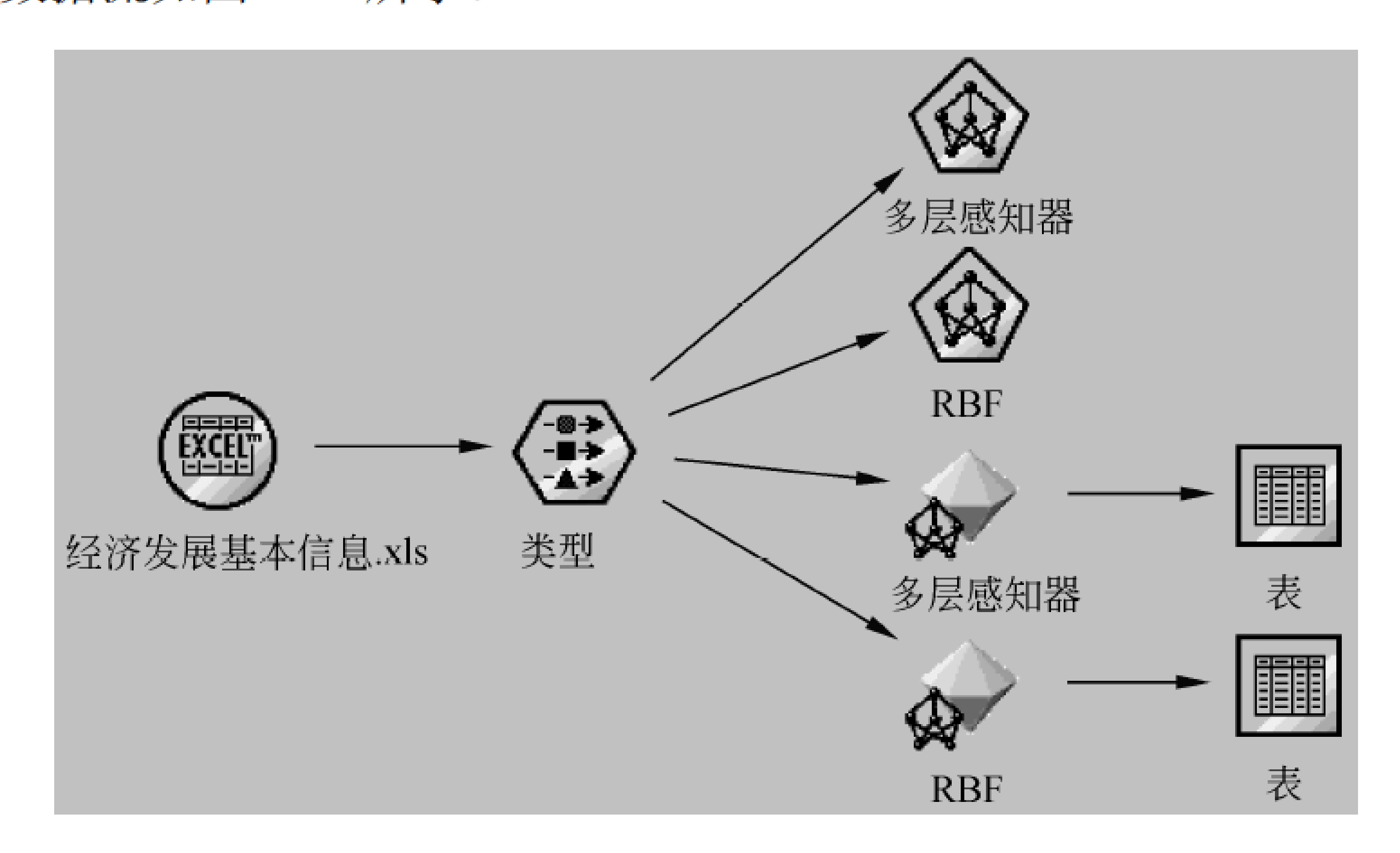


图 8.12　神经网络训练数据流

首先，将“数据源”中的 Excel 节点添加到数据流区域，并将“经济发展基本信息.xls”文件加载到该节点。

向数据流中添加“类型”节点，并建立从“经济发展基本信息.xls”节点到“类型”节点的连接。打开“类型”节点的编辑窗口。将“地区”字段的方向设置为“无”，即该字段不参与建模，GDP 方向设置为“输出”，其他字段的方向为“输入”。然后单击“读取值”按钮，如图 8.13 所示。



图 8.13　设置“类型”节点

首先构建一个多层感知器。向数据流中添加“神经网络”建模节点，建立从“类型”节点到该建模节点的连接，然后打开该节点的设置窗口进行设置，在“模型”标签下的设置如图 8.14 所示。

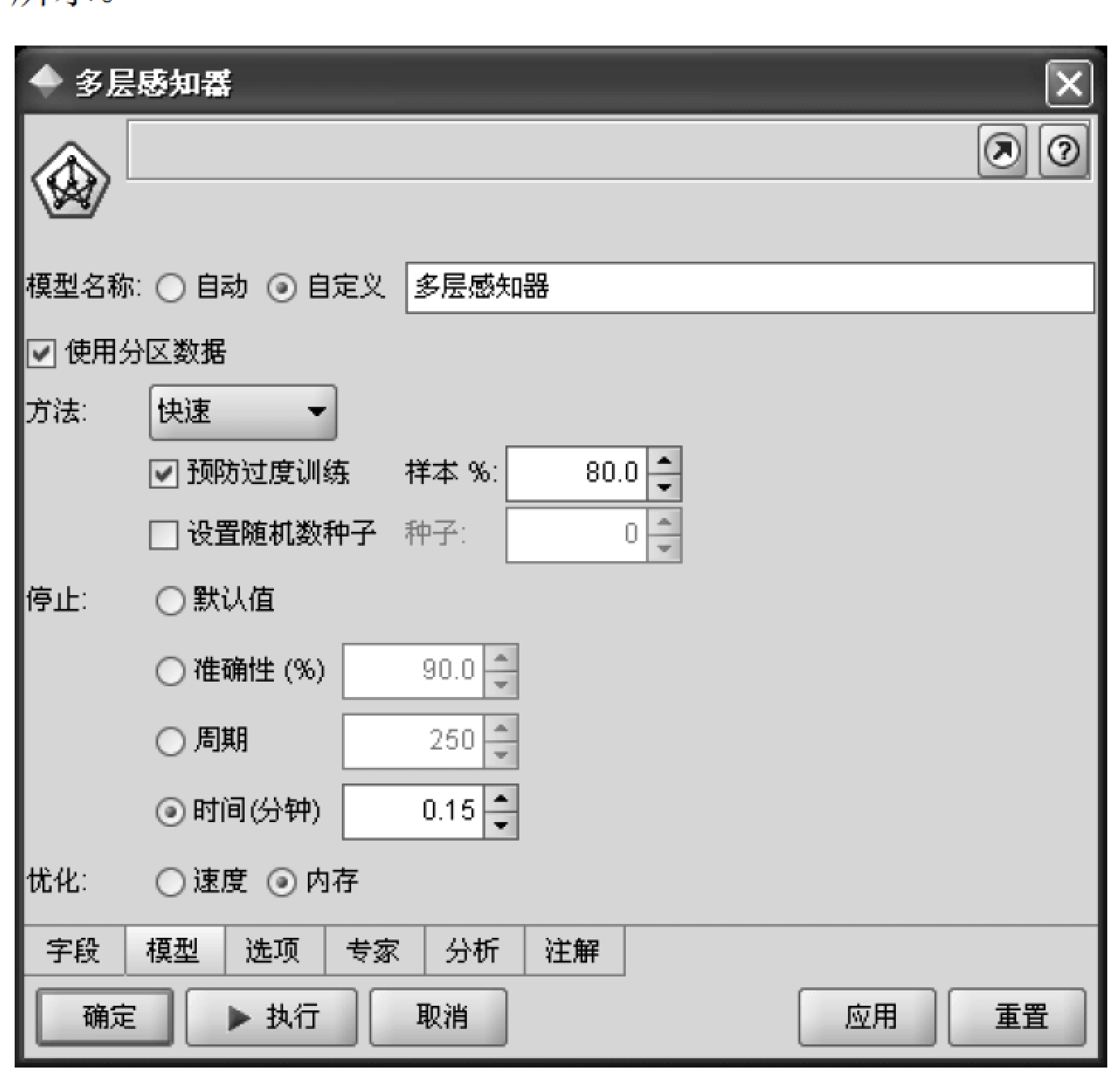


图 8.14　建模节点的基本设置

首先选中“自定义”模型名称，并在文本框中输入“多层感知器”，把该建模节点的名称改为“多层感知器”。

单击“方法”下拉列表框，可以看到一共有 6 种建模方法：快速、动态、多个、修剪、RBFN 和穷举型修剪。

（1）**快速方法（Quick Method）**。如果选择快速方法，在默认设置下，网络只含有一个隐藏层，层中含有 $\max(3,(n_i+n_o)/20)$个神经元，其中 n_i 是输入节点的数量，n_o 是输出神经元的数量。网络训练采用误差反向传播算法。如果在默认设置下生成的神经网络预测精确性较差，可以尝试在“专家”选项卡上增加隐藏层的大小，或者尝试其他训练方法。

（2）**动态方法（Dynamic Method）**。如果选择动态方法，在训练过程中首先创建一个最简单的初始拓扑，然后尝试着不断将神经元添加到网络中以提高网络性能，直到达到预期的精确性，以寻找一个最优的网络拓扑结构（即确定合适的隐藏层数以及各隐藏层包含合适的神经元数量）。最后再对这个最优的网络进行训练。

（3）**多个方法（Multiple Method）**。如果选择多个方法，在训练过程中首先创建多个具有不同拓扑的网络（确切数量取决于训练数据）。然后这些网络将以伪并行的方式进行训练。在所有网络训练结束后，预测精确性最高的那个网络被作为最后的模型。在默认情况下（用户没有指定网络结构），Clementine 将采用如下算法来创建多个具有不

同拓扑的网络：

首先生成若干个网络，这些网络都只有一个隐藏层，这些网络的隐藏层包含的神经元数量分别为 3，4，7，12，…，每次的增量 Δ_i 在上一次增量 Δ_{i-1} 上加 2。例如第一个网络的隐藏层神经元数量为 3，第二个网络的隐藏层神经元数量为 4，所以 $\Delta_1=4-3=1$，那么 $\Delta_2=\Delta_1+2=3$，所以第三个网络的隐藏层神经元数量为 $4+\Delta_2=7$，以此类推，直到达到输入节点数量（如果输入节点数量小于 12，那么隐藏层神经元数量增加到 12 为止；如果输入节点数量大于 60，则隐藏神经元数量不超过 60 个）。

然后，根据每个单隐藏层网络生成一组双隐藏层网络。这组双隐藏层网络中的第一个隐藏层的神经元数量与对应的单隐藏层网络所含神经元数量相同，第二个隐藏层的神经元数量分别为 2，5，10，17，…，以此类推，每次的增量 Δ_i 在上一次增量 Δ_{i-1} 上加 2。

用户也可以不采用上面的这种默认的方法，自行指定网络结构，即在这里选择多重方法，然后到“专家”标签下去对“拓扑”进行设置。例如，要生成具有一个隐藏层，其中隐藏神经元分别为 10、12、14 和 16 个的网络，以及具有两个隐藏层，其中隐藏神经元分别为 10 个和 7～10 个的网络，可以通过输入“10 16 2; 10, 7 10”来指定。

（4）**修剪方法（Prune Method）**。这种方法与动态方法恰好相反。修剪方法从一个较大的网络开始，然后在训练过程中逐渐删除隐藏层和输入层中最差的节点，从而找到最优的网络结构。修剪方法通常速度较慢，但相对其他方法产生的结果要好。用户可以选择这种方法后再到“专家”标签下对修剪方式的各种参数进行详细的设置。

（5）**RBFN**。该选项用于构建径向基函数网络。

（6）**穷举型修剪（Exhaustive Prune Method）**。穷举型修剪是修剪方法的一个特例，其修剪方式的各种参数都是确定的，例如初始网络的隐藏层为两层，其中第一隐藏层含 30 个神经元，第二隐藏层含 20 个神经元。此方法通常速度最慢，训练时间可能很长，尤其对于较大数据集更是如此，但产生的结果往往最好。

本例中选择“快速”方法。

预防过度训练（Prevent Overtraining）。此选项会将数据随机分割为两部分（训练集合和检验集合），以便于建模。网络的训练基于训练集合进行，精确性基于检验集合进行估计。神经网络节点的“样本%”框用于指定训练的数据比例，其余数据将用于检验精确性。本例中，勾选“预防过度训练”复选框。

设置随机数种子（Set Random Seed）。如果不设置随机数种子，则每次执行节点时用于初始化网络突触权值的随机值的序列都会不同。这将导致即使节点设置和数据值都完全相同，节点也会在不同的运行中创建不同的模型。通过选择该选项，可以将随机种子设置为特定值，从而使结果模型具有精确的可再现性。特定的随机种子通常会生成相同的随机值序列，在这种情况下执行节点通常会产生相同的生成模型。本例中，不勾选此选项。

停止条件（Stop On）。该选项用于设定停止训练的判定标准，有以下几种设定方法：

（1）默认值（Default）。默认的停止条件是根据“持续次数”（Persistence）来决定何时停止对网络的训练。“持续次数”指在网络在训练中没有继续得到改善的情况下会持续训练的周期数。值越高越有助于网络避免局部最小值，但这样也会延长训练时间。“持续次数”可以由用户在“专家”标签下自行设定。

（2）准确性（Accuracy）。如果使用此选项，在训练过程中的每一轮（周期）结束后，都会评估网络当前的预测准确性，训练过程会一直继续，直到网络当前的预测准确性达到用户设定的准确性阈值。如果准确性一直无法达到阈值，用户可以随时中断训练，以截止到目前所达到的最佳准确性保存该网络。准确性是对给定数据集中的数据进行预测并得到正确结果的比例度量。如果勾选了“预防过度训练”复选框，那么网络准确性的计算是基于测试数据集的，否则，网络准确性的计算基于整个训练数据集。另外，准确性的计算还依赖于输出字段的类型：如果输出字段是离散型，准确性是预测值与实际值相符的记录数占数据集记录数的比例；如果输出字段是连续型，那么准确性的计算公式为：

$$\text{accuracy} = \sum_R \frac{1 - |t_r - o_r|}{\max(t) - \min(t)} / n_r$$

其中，R 指数据集中的全部记录；t_r 是记录 r 的输出字段的实际值；o_r 是网络对记录 r 的输出字段的预测值；max(t)和 min(t)分别是输出字段在数据集 R 中的最大值和最小值；n_r 是 R 中的记录数量。

如果有多个输出神经元，那么网络准确性就是所有输出神经元准确性的平均值。

（3）周期（Cycles）。如果使用此选项，训练将持续指定的周期数（传递数据的次数）。

（4）时间（Time）。如果使用此选项，训练将持续指定的时间长度（以分钟为单位）。本例中选择此选项，设定为 0.15 分钟，即 9 秒钟。

以上设置完成后，切换到“选项”标签下，进行如图 8.15 所示的设置。

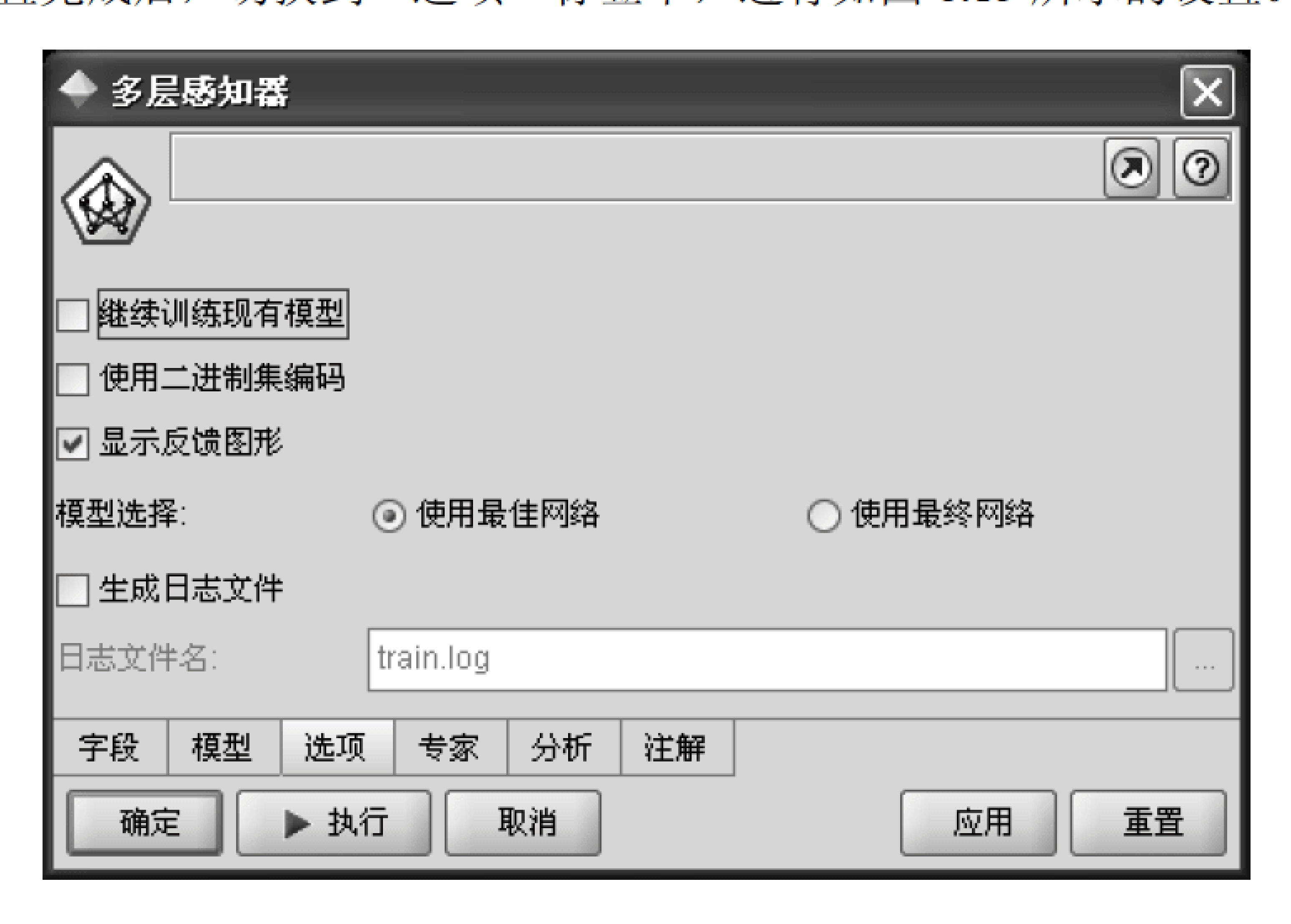


图 8.15 “选项”标签下的设置

在“选项”标签下，勾选“**显示反馈图形**”复选框，选择此选项，会在网络进行学习的同时，看到一个显示网络准确性变化情况的图形。例如本例在执行时生成的反馈图形如图 8.16 所示。

训练结束后的模型有两种选择：**使用最佳网络（Use Best Network）、使用最终网络（Use Final Network）**。默认情况下，训练结束后，节点会将最佳网络返回为生成的

网络节点。用户也可以要求该节点返回最终模型。本例选择“使用最佳网络”单选按钮。

神经网络“多层感知器”性能
100
0
最佳预测精确性 : 96.39 %
当前预测精确性 : 92.48 %

图 8.16　网络准确性的反馈图形

以上设置完成后，切换到“专家”标签下，进行如图 8.17 所示的设置。



图 8.17　多层感知器“专家”设置

首先将“模式”设置为“专家”。将网络的“隐藏层”设置为二层（隐藏层越多，越有助于神经网络了解更复杂的关系，但训练时间也会随之越长），每层所含的神经元数量均为 10（隐藏神经元越多，越有助于了解复杂的任务，但如同增加隐藏层一样，训练时间也会随之延长），“持续次数”为 200（值越高越有助于网络避免局部最小值，但这样也会延长训练时间）。

“学习速率”的各项参数设置均如图 8.17 中所示。各项参数的含义见第 8.2.1 小节。

以上设置完成后，单击“执行”按钮，即可显示如图 8.16 所示的反馈图形，待训练结束后，即可在管理器窗口的“模型”标签下显示生成的“多层感知器”网络模型。右

击该模型，在快捷菜单中选择“浏览”命令，在模型标签下显示变量重要性，可以看到“固定资产投资”对预测结果的影响最大。在“汇总”标签下显示汇总信息，如图 8.18 所示。

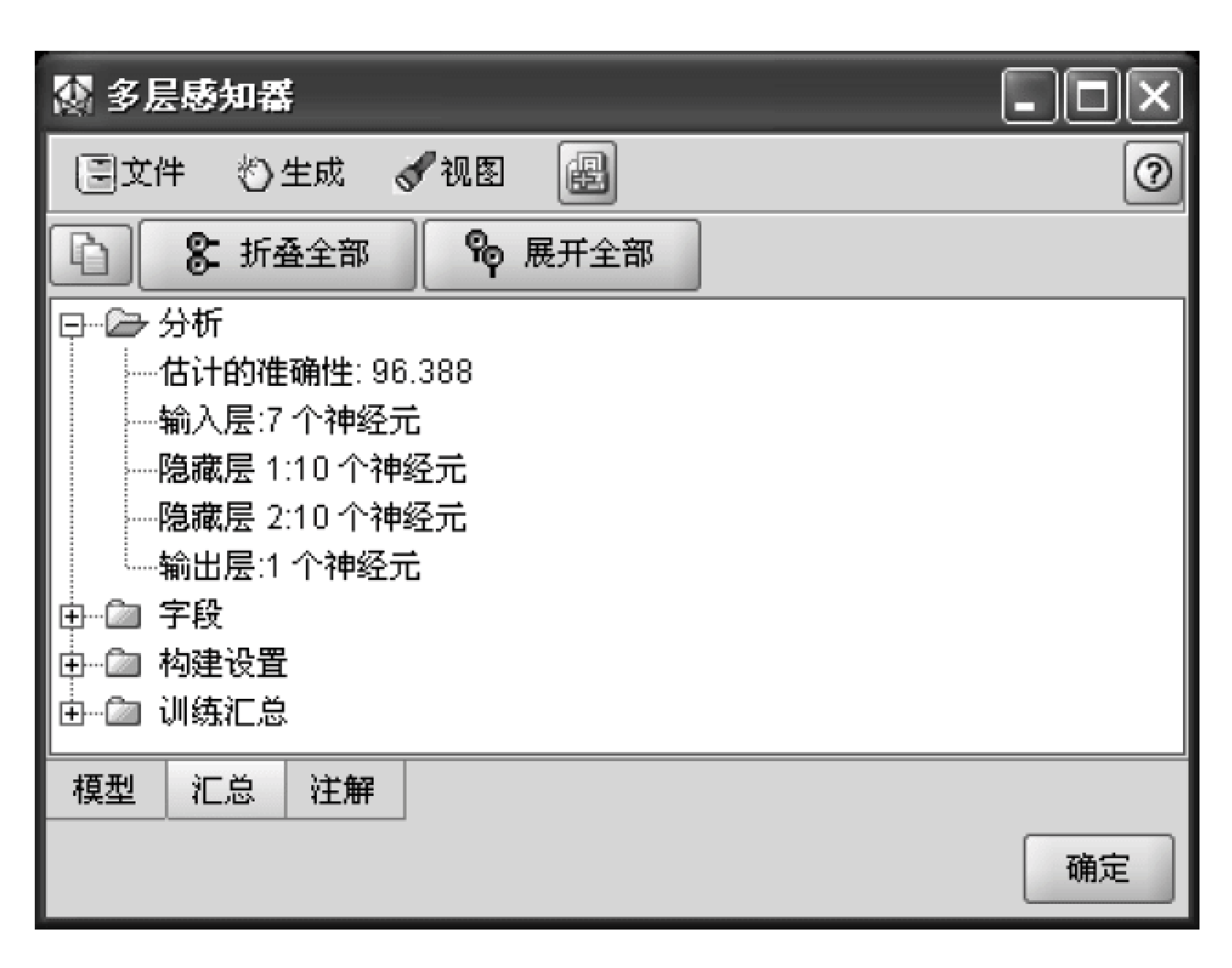


图 8.18　多层感知器模型汇总信息

可以看出，本次训练得到的神经网络预测准确性为 96.388%。Clementine 并没有为我们显示所得到的网络中的所有突触权值，不过这并不重要，因为我们无法对网络结构以及各突触权值的实际意义进行解释，这也正是神经网络的缺点之一。

注意，同样的设置下，再次进行同样的训练（即再次执行同一个神经网络建模节点），可能得到的结果并不完全相同。这是因为在训练开始时，网络中的所有权值都被随机地设置为一个属于[–0.5,0.5]的值，因此两次不同的训练中，初始权值往往是不同的。

下面再来训练一个 RBF 神经网络。向数据流中再添加“神经网络”建模节点，建立从“类型”节点到该建模节点的连接，然后打开该节点的设置窗口进行设置，在“模型”标签下，选中“自定义”模型名称，并在文本框中输入“RBFN”，把该建模节点的名称改为“RBFN”。

单击“方法”下拉列表框，选择 RBFN。勾选“预防过度训练”复选框，“样本%”设置为 80。停止条件选择“时间（分钟）”，设定为 0.15。

“选项”标签下的设置与前面多层感知器的设置相同。

切换到“专家”标签下，可以设置 RBFN 专家选项，如图 8.19 所示。

图中各参数的含义见第 8.2.2 小节。

以上设置完成后，单击“执行”按钮，即可显示反馈图形，待训练结束后，在管理器窗口的“模型”标签下显示生成的 RBFN 网络模型。同样，用户可以浏览该模型的信息。

将生成的两个神经网络模型添加到数据流区域，并建立从类型到这两个模型节点的连接，即可将模型用于对整个数据集的预测，在模型节点后面添加“表”节点并执行，即可显示预测结果。多层感知器的预测结果如图 8.20 所示。



图 8.19 RBFN 专家设置


表（10 个字段，30 条记录）

文件 编辑 生成

	均工资	货物...	居民消费价...	商品零售价...	工业总...	GDP	$N-GDP
1	4.000	373.9...	117.300	112.600	843.430	1394.890	1613.522
2	1.000	342.8...	115.300	110.600	582.510	920.110	887.572
3	9.000	2033....	115.200	115.800	1234.8...	2849.520	3092.532
4	1.000	717.3...	116.900	115.600	697.250	1092.480	1326.008
5	4.000	781.7...	117.500	116.800	419.390	832.880	1129.492
6	1.000	1371....	116.100	114.000	1840.5...	2793.370	3081.111
7	0.000	497.4...	115.200	114.200	762.470	1129.200	1366.183
8	5.000	824.8...	116.100	114.300	1240.3...	2014.530	1754.153
9	9.000	207.4...	118.700	113.000	1642.9...	2462.570	2667.412
10	3.000	1025....	115.800	114.300	2026.6...	5155.250	5217.210
11	9.000	754.4...	116.600	113.500	916.590	3524.790	3781.041
12	9.000	908.3...	114.800	112.700	824.140	2003.580	2273.373
13	7.000	609.3...	115.200	114.400	433.670	2160.520	2114.773

表 注解 确定

图 8.20 多层感知器预测结果

图中，列$N-GDP 即为预测结果。

8.3 Kohonen 网络

Kohonen 网络是一种自组织神经网络，是无监督学习的神经网络模型，采用了竞争型的学习机制，可以对外界未知环境进行学习或模拟，并对自身网络结构进行适当的调整。

8.3.1 自组织神经网络

虽然多层感知器、RFBN 等前向网络得到了广泛的应用，但其实它们并没有充分借

鉴人脑的工作特点，因而其功能还有许多不足之处。例如，前向网络在训练过程中所有权值都要调整，存在稳定性及学习速度等问题，还可能陷入均方误差的局部最小点，从而造成错误的求解结果，另外，前向网络往往只适用于平稳的环境，不能随环境的变化而进行相应的调整或改变。

其实，人脑可以在一个复杂的、有干扰的环境中辨识学习的目标并学习大量知识，这种学习方式是自主的，是不受导师明确指导的"自学"。同时，人脑在学习过程中是有所侧重的，有些知识会学习并记忆得很牢，也能够忽略或忘记那些无关紧要的信息。人能够专注于某些客体或者某些关系，而置其他于不顾或者不给予足够的重视。

根据人脑的这些特点，人们研究了更加接近于人脑工作特性的人工神经网络模型，这就是自组织神经网络，也称为竞争神经网络。

自组织神经网络是以自组织学习方式进行学习的，其目的是在无导师指导下寻找输入数据空间中的重要的模式或特征。自组织学习的网络结构模型，比有监督学习的网络结构模型更加接近生物神经系统，因为网络的自组织过程正是人脑组织的一个基本现象。

自组织学习，也称竞争学习，是指在同一层上的神经元之间相互进行竞争，只有竞争胜利的神经元才修改与其相连的权值。这种机制可以用来进行模式分类。竞争学习是一种无监督的学习。在无监督学习中，只向网络提供一些学习样本，而没有期望输出。网络根据输入样本进行自组织，并将其划分到相应的模式类别中。由于不需要提供理想输出，因而推广了有监督模式分类方法。

基本的自组织学习网络由两个层次组成，即输入层和竞争层（输出层）。在竞争层中，神经元之间相互竞争，最终只有一个或几个神经元活跃，以适应当前的输入样本。竞争胜利的神经元就代表着当前输入样本的分类模式。

以竞争神经网络为基础，可以构成一些具有自组织能力的神经网络，如自适应共振理论（Adaptive Resonance Theory，ART）网络、自组织特征映射（Self-Organizing Feature Map，SOM）网络、主分量分析（Principle Components Analysis，PCA）网络、对传（Counter Propagation，CP）网络和协同神经网络（Synergetic Neural Network，SNN）等。其中，自组织特征映射网络就是本节要介绍的 Kohonen 网络。

8.3.2 自组织特征映射网络

自组织特征映射网络也称为 Kohonen 网络，由芬兰学者 Teuvo Kohonen 于 1981 年提出。

1. 基本原理

Kohonen 认为，处于空间中不同区域的神经元有不同的分工，当一个神经网络接受外界输入模式时，将会分为不同的反应区域，各区域对输入模式具有不同的响应特征。另外，神经元的响应会对其周边的其他神经元产生交互的影响，也就是相邻近的神经元之间的局部交互，其交互方式为侧向交互。这种侧向交互方式遵从下列有关规则：

（1）以发出信号的神经元为圆心，对近邻的神经元的交互作用表现为兴奋性侧反馈。

（2）以发出信号的神经元为圆心，对远邻的神经元的交互作用表现为抑制性侧反馈。

这种规则说明近邻者相互激励，远邻者相互抑制。一般而言，近邻是指从发出信号的神经元为圆心，半径约为 50～500μm 左右的神经元；远邻是指半径为 200μm～2mm 左右的神经元。比远邻更远的神经元则表现的是弱激励作用。这样，这种局部交互方式用函数形式来表示如图 8.21 所示。由于这种交互作用的曲线类似于墨西哥人戴的帽子，所以也称为“墨西哥草帽函数”。

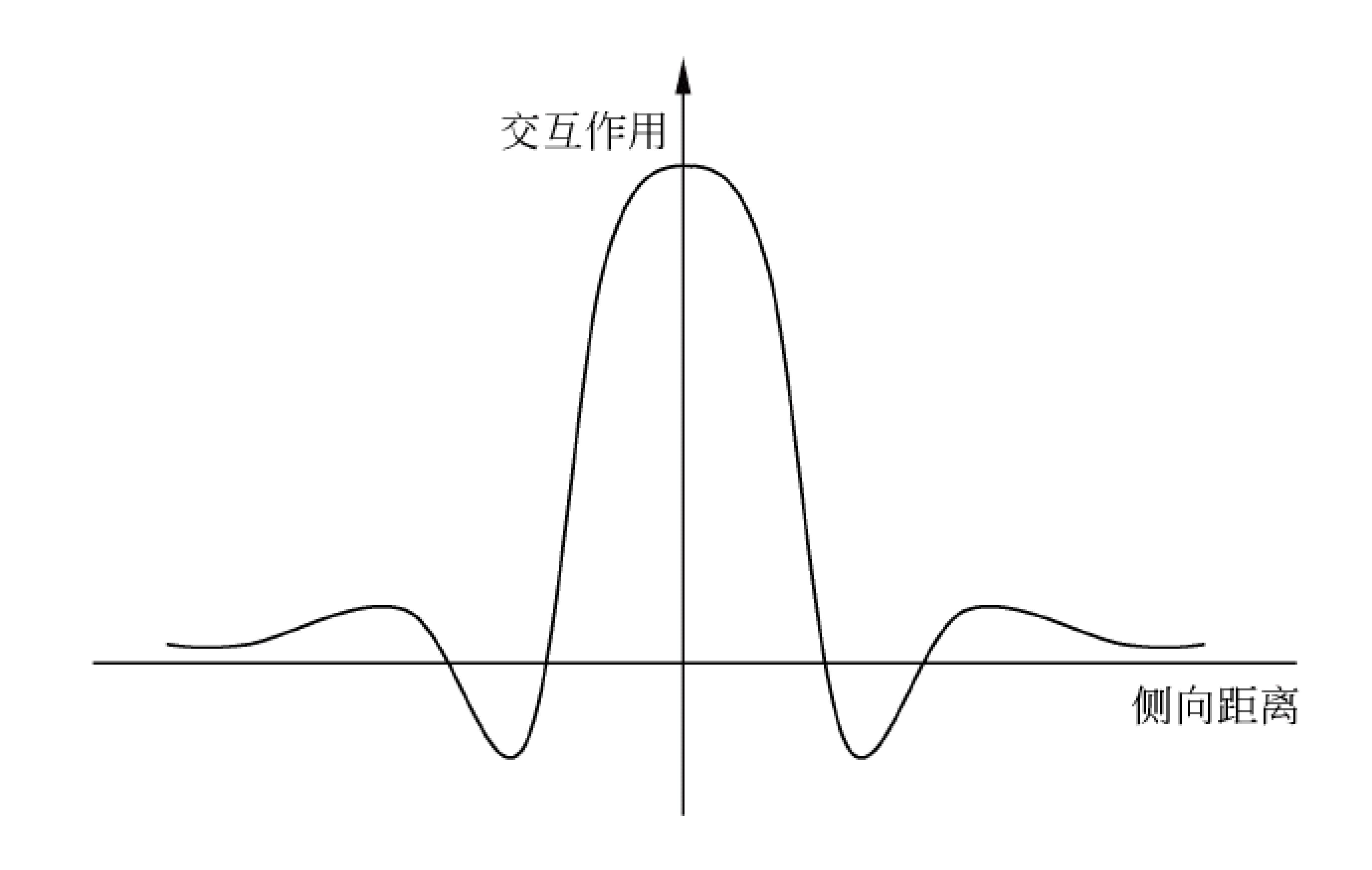


图 8.21　墨西哥草帽函数

神经网络中，邻近的各个神经元之间通过相互的侧向交互作用，从而产生相竞争的过程，自适应地形成了针对特殊信息的组织结构。很明显，这种结构可以成为检测特殊信息的特殊检测器。这样，神经网格的自组织过程和行为，可以成为一种检测各种不同信息模式的检测相识别器。这也是自组织特征映射的意义所在。

2．Kohonen 网络结构

Kohonen 网络可以用二维阵列表示，这种结构如图 8.22 所示。

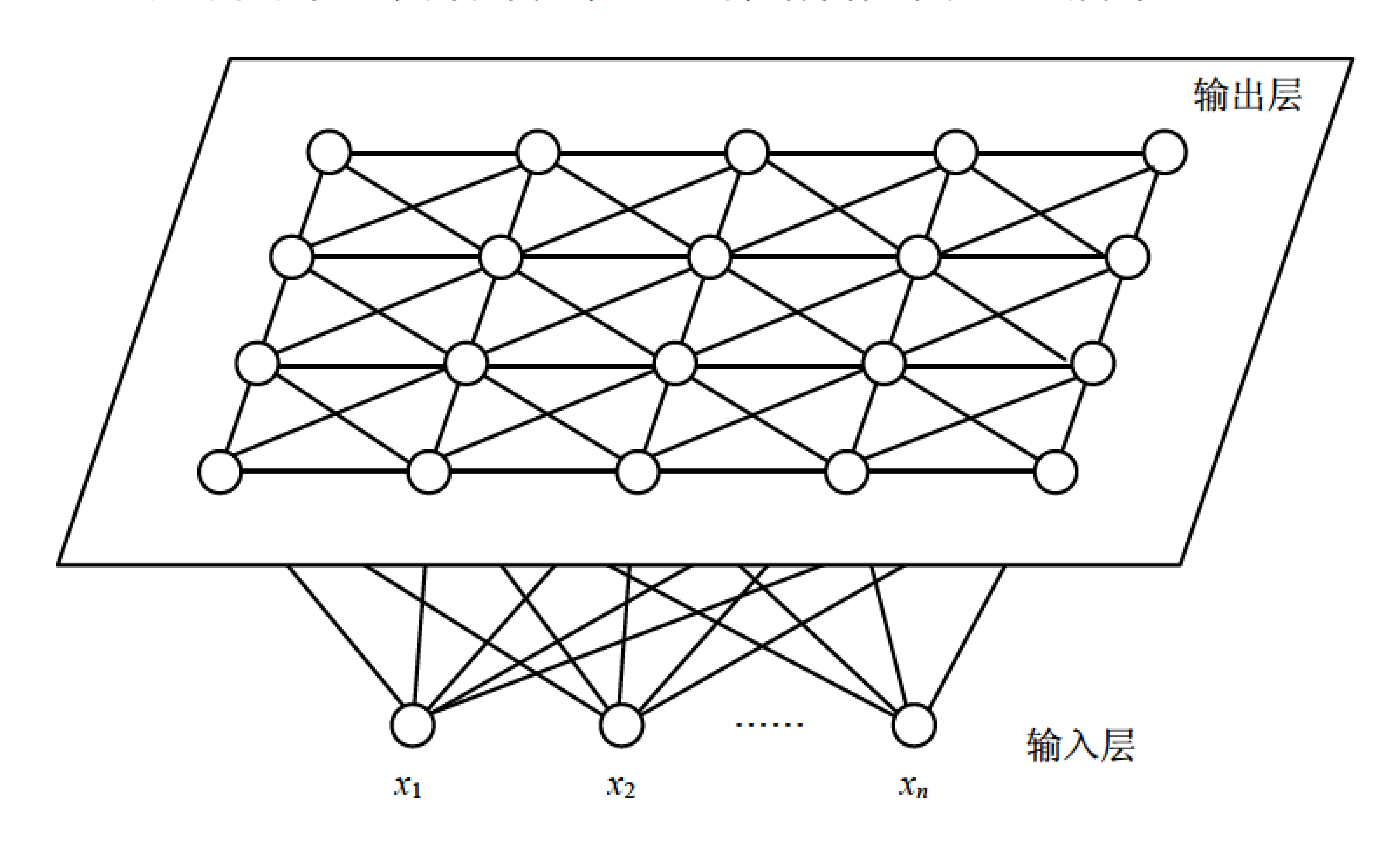


图 8.22　二维阵列 Kohonen 网络

Kohonen 网络由输入层和竞争层组成。输入层是一维的神经元，竞争层是二维的神经元。一个输入层的神经元和竞争层的所有神经元都相互连接（全连接），每个连接都有一个相应的权值。竞争层也称输出层。

在竞争层中，每个输出神经元都和最近相邻的 8 个神经元相连；当然，最边沿的神经元和 3～5 个神经元相连，但这只是最边沿的神经元才会这样。

设输入层由 K 个神经元组成，输出层由 J 个神经元组成，一个输入层神经元 k 与一个输出层神经元 j 的连接权值记为 w_{jk}。

3．Kohonen 网络学习算法

在网络训练之初，先将所有权值初始化为一个小的随机值。

下面的学习过程分为 3 个步骤：竞争、确定邻域、更新权值。

（1）竞争

竞争的过程就是在输出层神经元中确定获胜神经元的过程，输出最大的神经元就是获胜神经元。由于输出层神经元的激励函数为线性函数，其最大输出取决于其输入。

当输入样本 i（含 K 个字段）时，输出神经元 j 的输入为：$u_j=\sum_{k=1}^{K}x_{ik}w_{jk}$，其中，$x_{ik}$ 是输入层的神经元 k 的输入（即样本 i 的第 k 个字段的值），如图 8.23 所示。

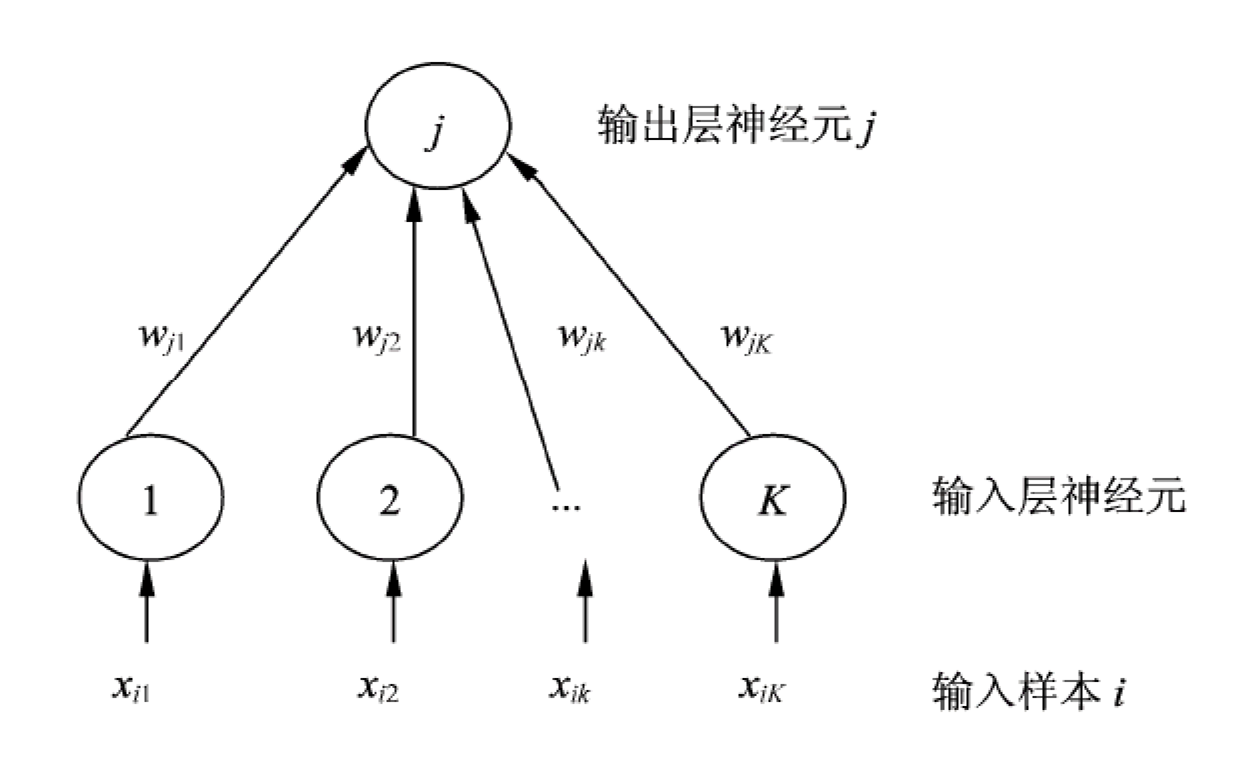


图 8.23　输出层神经元的输入

显然，u_j 也可以表示为输入向量 $X_i=[x_{i1},x_{i2},\cdots,x_{iK}]^T$ 与权值向量 $W_j=[w_{1k},w_{2k},\cdots,w_{Jk}]^{\mathrm{T}}$ 的内积，等价于输入向量和权值向量的欧氏距离最小。当输入样本 i 时，输出神经元 j 的输入向量和权值向量的欧氏距离为：

$$d_{ij}=\sqrt{\sum_{k}(x_{ik}-w_{jk})^2}$$

计算输出层 J 个神经元的这个欧氏距离，距离最小的那个神经元即为获胜神经元。

（2）确定邻域

在上一步骤中已经确定了获胜神经元，这里 3 把获胜神经元称为“加强中心”。现在，要确定一个以“加强中心”为中心的邻域。与前面介绍的前向网络不同，Kohonen 网络在训练过程中只更新邻域中的神经元的权值，而不是更新所有神经元的权值。

通常采用正方形的邻域形状。当邻域半径为 0 时，邻域仅仅包含获胜神经元。当邻域半径为 1 时，邻域包含了获胜神经元以及与之相邻的 8 个神经元。当半径增大时，邻域包含的神经元逐渐增多。

设获胜神经元为 C，在第 n 次迭代时的邻域半径记为 $N_C(n)$。邻域半径的值随着迭代次数的增加而变化，其变化规则为

$$N_C(n) = \mathrm{INT}(N_C(0)(1-n/N))\text{，}\quad n = 0,1,2,\cdots,N$$

其中，$N_C(0)$表示初始设置的邻域半径，N 为总的迭代次数，INT 为取整函数。显然，邻域随着迭代次数的增加而不断变小。图 8.24 显示了邻域 N_C 随迭代次数的变化情况。

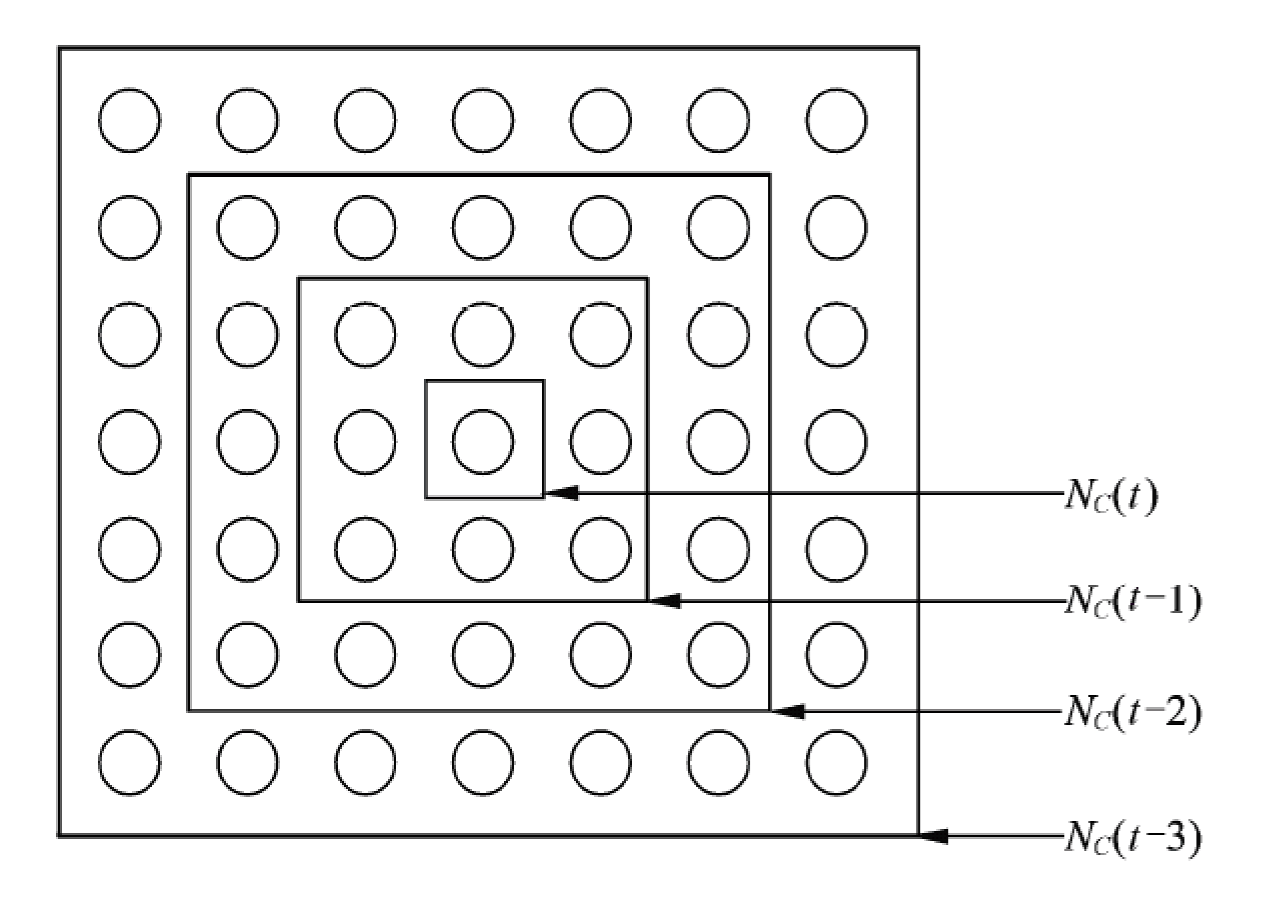


图 8.24　邻域 N_C 随迭代次数变化

从图中可以看出 $N_C(t-1)$的范围比 $N_C(t-2)$要小；当到达次数 t 时，邻域 $N_C(t)$则处在神经元 C 处。

在 Clementine 中，对于获胜神经元 o_i，如果其邻近的神经元 o_j 满足 $d_C(o_i,o_j)<r$，r 是邻域半径，就认为神经元 o_j 在以 o_i 为中心的邻域内，这里，d_C 是基于切比雪夫距离（Chebychev Distance）的邻域函数：

$$d_C(x,y) = \max_i |x_i - y_i|$$

代表了神经元 x 和神经元 y 在任一维度 i 上的最大距离，其中，x_i 是神经元 x 在维度 i 上的位置，y_i 是神经元 y 在维度 i 上的位置。

（3）更新权值

对于获胜神经元及其邻域内的神经元，它们的权值要被更新。例如邻域内的神经元 j，在第 t+1 次迭代中，更新前的权值向量为 $W_j(t)$，更新后的权值向量为

$$W_j(t+1) = W_j(t) + \Delta W_j(t)$$

其中，$\Delta W_j(t) = \eta(W_j(t) - I_j(t))$，$\eta$ 是学习率参数，$I_j(t)$ 是神经元 j 的输入向量。

在每一次迭代结束时，学习率参数 η 的值也被更新。在训练过程中，η 的值随迭代次数的增加而递减，变化的规则可以采用下式：

$$\eta(t) = \eta(0)\left(1 - \frac{t}{c}\right)\text{，}\quad t = 0,1,2,\cdots,c$$

式中，$\eta(0)$ 为初始学习速率（由用户设定），c 为总迭代次数，t 为当前迭代次数。

在 Clementine 中则有两种递减的方式可以选择：

① 线性递减。Clementine 默认的递减方式就是线性递减。在这种方式下，η 每次递减一个固定的数值，计算公式为

$$\eta(t+1)=\eta(t)-\left(\frac{\eta(0)-\eta_{\text{low}}}{c}\right)$$

其中，$\eta(0)$ 为初始学习速率，η_{low} 代表 η 的最小值，也就是 η 递减到 η_{low} 后，迭代停止。

② 指数递减。在这种方式下，η 每次递减一个固定的比例，计算公式为：

$$\eta(t+1)=\eta(t)\cdot\exp\left(\frac{\log\left(\frac{\eta_{\text{low}}}{\eta(0)}\right)}{c}\right)$$

这里，η_{low} 的最小值为 0.0001，以避免在进行对数运算时发生算术错误。

8.3.3 在 Clementine 中应用 Kohonen 网络

在 Clementine 中，Kohonen 网络被用于聚类分析，因此 Clementine 将其划分到了聚类方法中。但由于涉及了神经网络的知识，所以将 Kohonen 网络的应用放在这里来介绍。

本节在 Clementine 中用 Kohonen 网络对第 4.2.3 节中的液体饮料聚类分析实例进行数据挖掘。样本数据集在表 4-3 中。

用 Kohonen 网络对液体饮料进行聚类分析的数据流如图 8.25 所示。

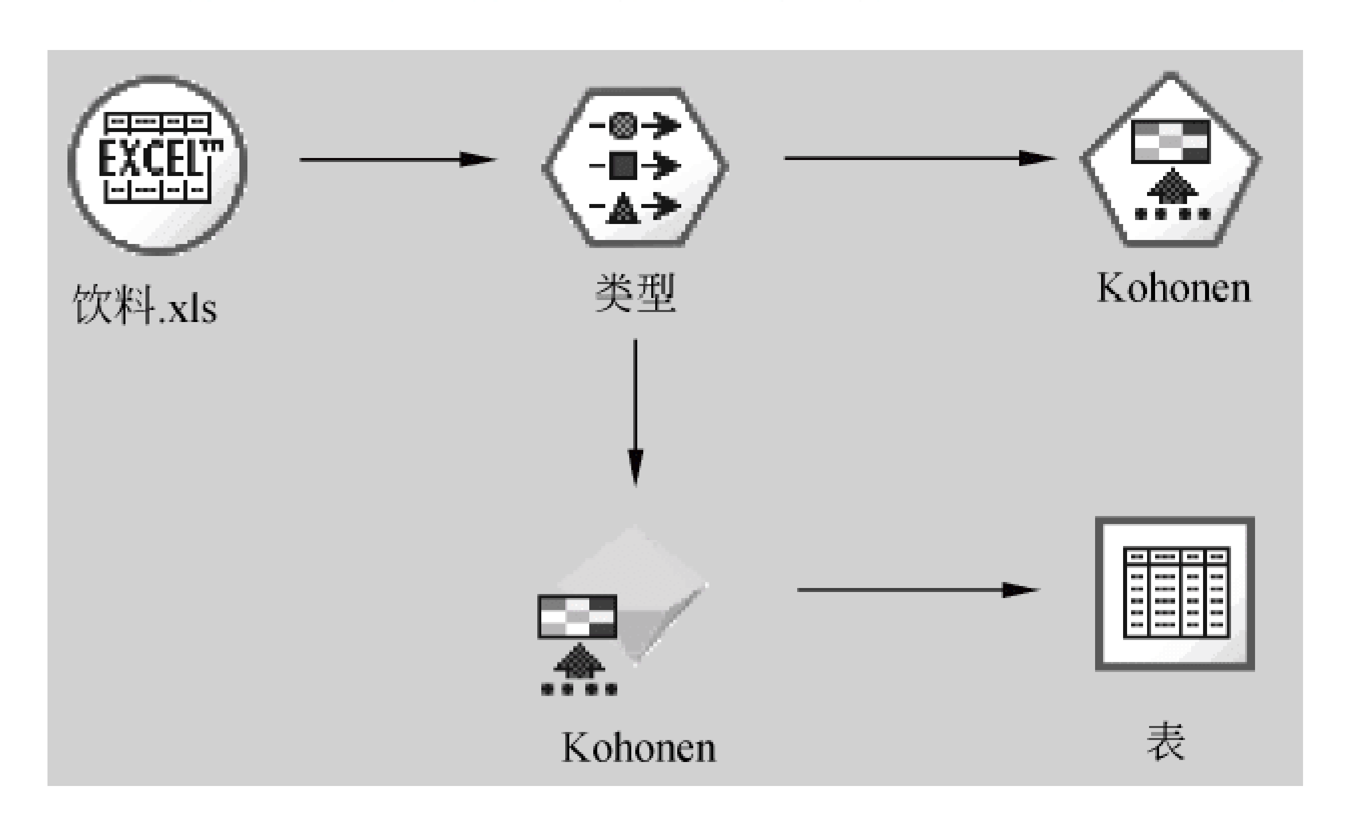


图 8.25　Kohonen 网络聚类分析数据流

首先在数据流区域添加 Excel 数据源节点并导入“饮料.xls”文件，在该节点将“编号”属性过滤。然后添加“类型”节点，并读取值。注意，在“类型”节点无需指定输出字段（所有字段均为输入字段），因为 Kohonen 网络的训练是一种无监督学习，不用于预测目标字段，而是试图揭示输入字段集中的模式。

在数据流区域添加 Kohonen 节点，并建立从“类型”节点到 Kohonen 节点的连接，然后编辑 Kohonen 节点，如图 8.26 所示。

在“模型”标签下，勾选“显示反馈图形”复选框，选择此选项，会在训练期间显

示二维数组的直观表示。每个节点（代表一个输出层的神经元）的强度用颜色表示。红色表示聚集了许多记录的单元（强单元），白色表示聚集的记录较少或没有记录的单元（弱单元）。注意，此功能会减慢训练进度。要加快训练进度，可取消选中此复选框。例如本例在执行时生成的反馈图形如图 8.27 所示。



图 8.26　Kohonen 网络模型基本设置



图 8.27　Kohonen 网络反馈图形

停止条件（Stop On）有两个选项：默认、时间。在默认情况下，网络训练将在完成指定迭代次数之后停止。如果选择“时间”，网络训练将在指定时间长度（分钟）之后

停止（或者尚未达到指定时间长度但已完成指定迭代次数也将停止）。

切换到“专家”标签下，选择“专家”模式。相应设置如图 8.28 所示。



图 8.28　Kohonen 网络模型高级设置

对“宽度”和“长度”进行设置。将二维输出图的大小（宽度和长度）指定为每个维上的输出单元数，实际上指定了输出层神经元的数量，这里设置宽度为 3，长度为 2，所以输出层神经元数量为 6。

学习速率衰减（Learning Rate Decay）选择“线性”（Linear）。

阶段 1 和阶段 2（Phase 1 and Phase 2）：Kohonen 网络训练分为两个阶段。阶段 1 是粗略估计阶段，用于捕获数据中的大致模式。阶段 2 是调整阶段，用于调整图以便为数据更精细的特征建模。每个阶段都有以下 3 个参数：

（1）**近邻（Neighborhood）**。这里的近邻也就是前文中讲到的“邻域”，这里该参数设置的是邻域半径的起始大小。

在阶段 1，邻域半径从起始大小（即[阶段 1 近邻]，本例中设置为 2）开始，随迭代次数逐渐减小，直到减小到（[阶段 2 近邻]+ 1）。然后在阶段 2，邻域半径起始为[阶段 2 近邻]，然后减少到 1.0。[阶段 1 近邻]应大于[阶段 2 近邻]。

（2）**初始 Eta（Initial Eta）**，即 $\eta(0)$。在阶段 1，$\eta(0)$ 为[阶段 1 初始 Eta]（这里设置为 0.3），开始训练后，η 逐渐减小到[阶段 2 初始 Eta]。在阶段 2，$\eta(0)$ 为[阶段 2 初始 Eta]，然后减少到 0。[阶段 1 初始 Eta]应大于[阶段 2 初始 Eta]。

（3）**周期（Cycles）**，为训练的每个阶段设置迭代次数。每个阶段均会进行指定次数的数据处理。

以上设置完毕后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示生成的 Kohonen 模型。右击生成的 Kohonen 模型节点，在快捷菜单中选择“浏览”命令，打开 Kohonen 对话框，在模型标签下（如图 8.29 所示）可以看到，Kohonen 网络将 15 个样本划分为了 3 个聚类，单击“全部展开”按钮，则可以显示每个聚类的一些统计

信息。



图 8.29　聚类的统计信息

例如第一个聚类，由 X=0、Y=0 所代表的那个神经元来表示（X 和 Y 是输出层网格上的坐标轴），该聚类包含 4 个样本。

单击“查看器”标签，可以以图表的形式来显示模型的统计信息以及各个属性在各聚类中的分布信息，如图 8.30 所示。

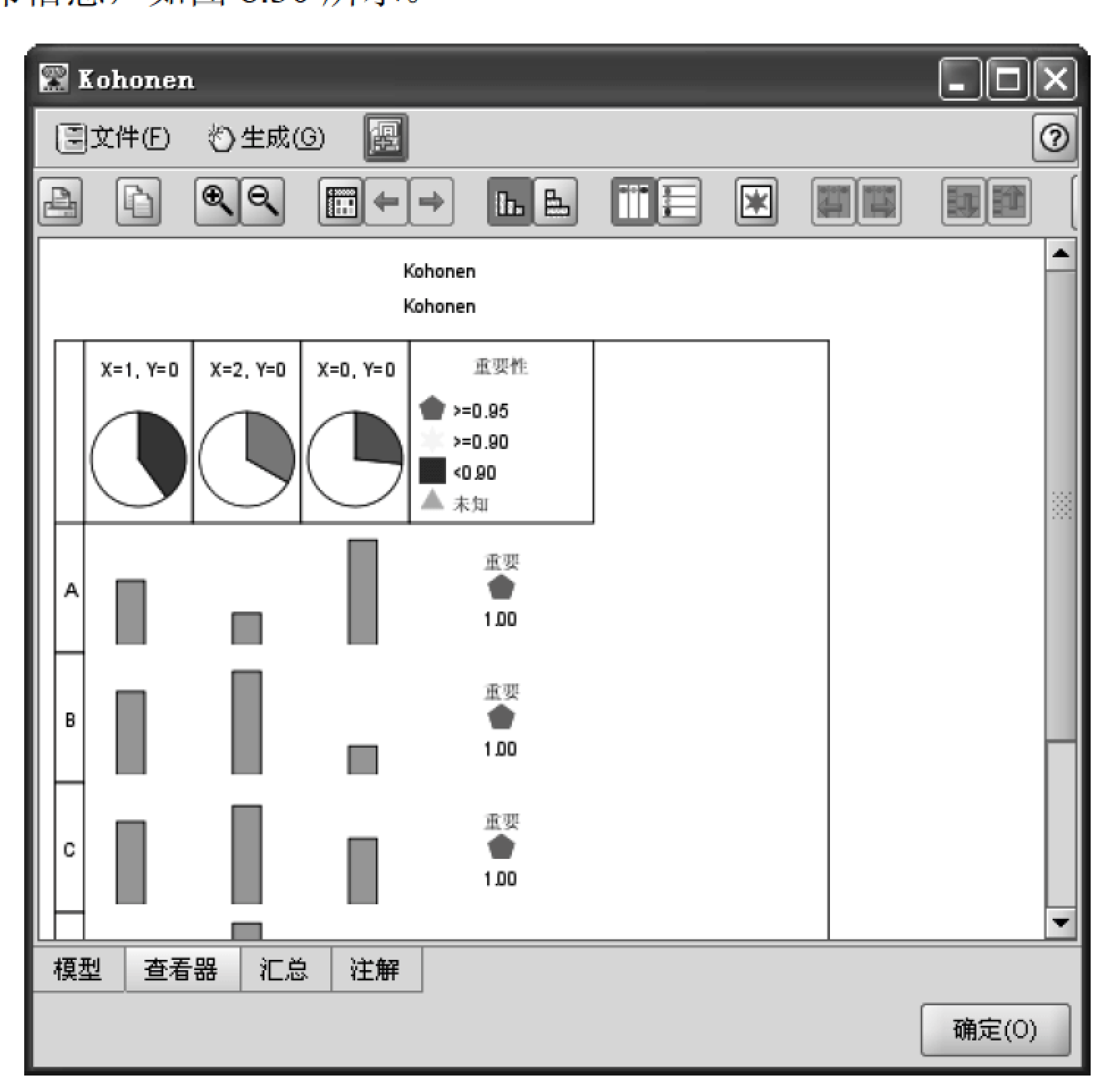


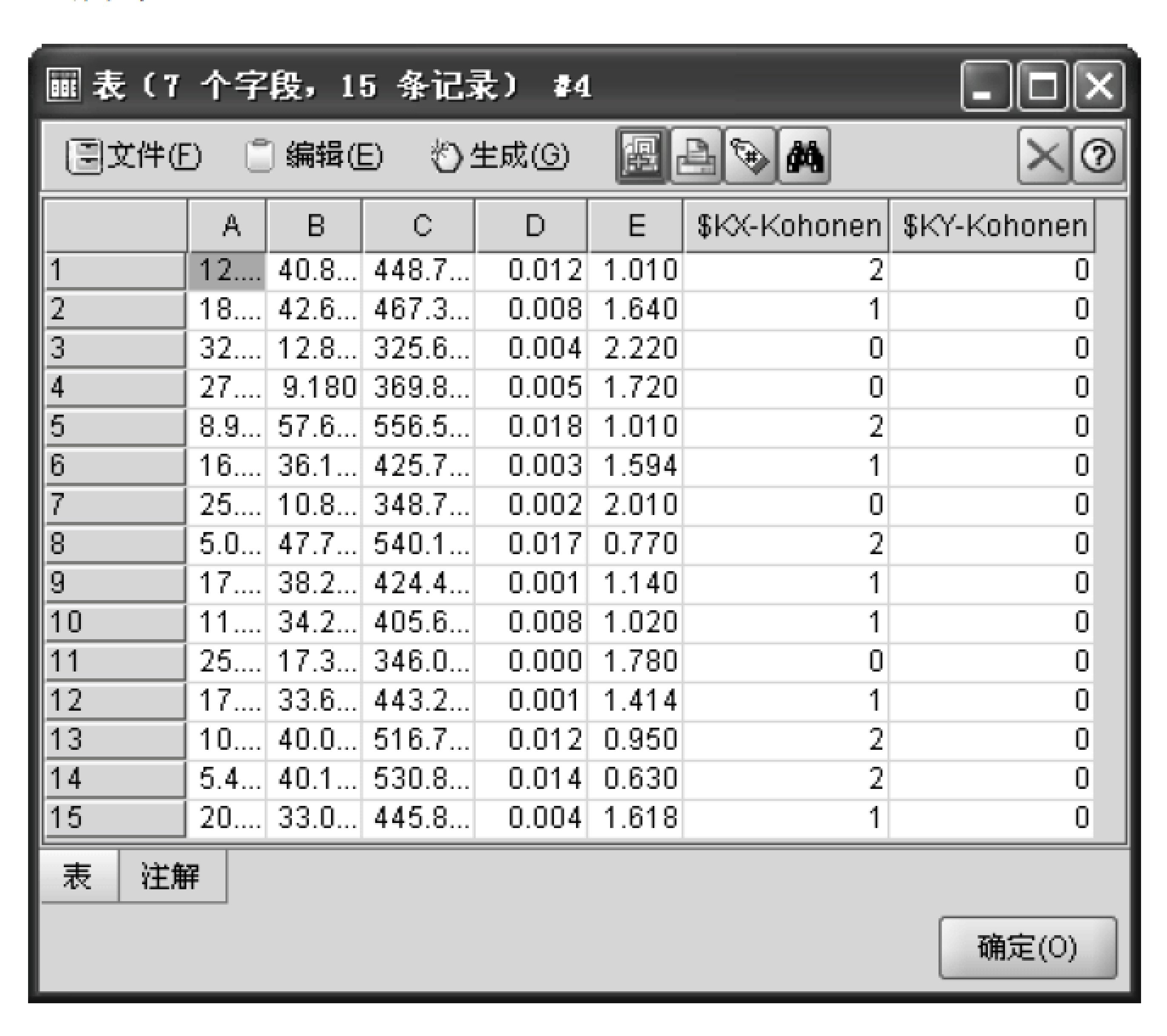
图 8.30　模型的“查看器”标签

图中各元素所代表的含义请参考第 4.2.3 节。

最后，在“汇总”标签中，会显示关于本次训练过程中的一些概要信息。

将生成的 Kohonen 模型拖入到数据流区域，并建立从“类型”节点到该节点的连接，

然后在模型节点后添加“表”节点，执行“表”节点，即可看到对样本数据集进行聚类的结果，如图 8.31 所示。



	A	B	C	D	E	$KX-Kohonen	$KY-Kohonen
1	12....	40.8...	448.7...	0.012	1.010	2	0
2	18....	42.6...	467.3...	0.008	1.640	1	0
3	32....	12.8...	325.6...	0.004	2.220	0	0
4	27....	9.180	369.8...	0.005	1.720	0	0
5	8.9...	57.6...	556.5...	0.018	1.010	2	0
6	16....	36.1...	425.7...	0.003	1.594	1	0
7	25....	10.8...	348.7...	0.002	2.010	0	0
8	5.0...	47.7...	540.1...	0.017	0.770	2	0
9	17....	38.2...	424.4...	0.001	1.140	1	0
10	11....	34.2...	405.6...	0.008	1.020	1	0
11	25....	17.3...	346.0...	0.000	1.780	0	0
12	17....	33.6...	443.2...	0.001	1.414	1	0
13	10....	40.0...	516.7...	0.012	0.950	2	0
14	5.4...	40.1...	530.8...	0.014	0.630	2	0
15	20....	33.0...	445.8...	0.004	1.618	1	0

图 8.31　Kohonen 模型的聚类结果

可以看到，在聚类结果中有两列数据：$KX-Kohonen、$KY-Kohonen，分别是输出层网格上的 *X* 轴坐标和 *Y* 轴坐标。

所以，代表 3 个聚类的神经元的坐标以及所含样本分别为：

聚类 1：(2, 0)　　含 5 个样本：1、5、8、13、14

聚类 2：(1, 0)　　含 6 个样本：2、6、9、10、12、15

聚类 3：(0, 0)　　含 4 个样本：3、4、7、11

第 9 章　时间序列分析与预测

9.1 时间序列概述

时间序列分析（Time series analysis）是一种广泛应用的数据分析方法，主要用于描述和探索现象随时间发展变化的数量规律性。近年来，时间序列挖掘在宏观经济预测、市场营销、金融分析等领域得到应用。时间序列分析通过研究信息的时间特性，深入洞悉事物发展变化的机制，成为获得知识的有效途径。

9.1.1 时间序列基本概念

时间序列是一种常见的数据形式。同一现象在不同时间的相继观察值排列而成的序列，称为时间序列。根据观察时间的不同，时间序列中的时间可以是年份、季度、月份或其他任何时间形式。用 t 表示所观察的时间，$Y(t)$表示在时间 t 的观察值。

时间序列在长时期内呈现出来的某种持续增长或者持续下降的变动，称为趋势（trend）。

时间序列可以分为平稳序列和非平稳序列。其中，基本上不存在趋势的序列，称为“平稳序列”（Stationary Series）；包含趋势性、季节性或周期性的序列，称为“非平稳序列”（Non-stationary Series）。

平稳序列的各观察值基本上在某个固定的水平上波动，虽然在不同的时间段波动的程度不同，但并不存在某种规律，其波动可以看成是随机的。非平稳性序列则可能包含着某种趋势，或者多种趋势的组合。

对于非平稳序列来说，其趋势是由于某种固定性的因素作用于序列而形成的，其趋势可以是线性的，也可以是非线性的。

如果时间序列在一年内重复出现周期性的波动，则称该序列具有季节性（seasonality）。注意，这里的“季节”含义是广义的。它不仅仅指一年中的四季，而是泛指任何一种周期性的变化，如“旅游旺季”、“销售淡季”等。

如果时间序列长期（而非一年）呈现出一种波浪形或振荡式变动，则称该序列具有周期性（cyclity）。周期性变动不同于趋势变动，不是朝着一个方向持续增大或减小，也不同于季节变动，季节变动有比较固定的规律，且变动周期大多为一年，周期性变动没有固定规律，变动周期多在一年以上，且周期长短不一。

另外，有些偶然因素也对时间序列产生影响，导致时间序列呈现出某种随机波动，称为随机性（random）。

对于平稳序列以及各种具有不同特性的非平稳序列，对它们的分析方法各不相同。因此，在进行序列分析之前，有必要根据数据先作出散点图或采用其他分析方法，分析

这个序列是平稳序列还是非平稳序列（以及具有何种特性的非平稳序列），然后选择合适的分析方法。

9.1.2 时间序列预测的传统方法

1. 简单平均法

根据过去已有的 t 期观察值通过简单平均来预测第 t+1 期的数值，这种预测方法称为简单平均法。

设时间序列已有的 t 期观察值为 $Y(1), Y(2), \cdots, Y(t)$，那么对第 t+1 期的预测值为：

$$\hat{Y}_{t+1}=\frac{1}{t}[Y(1)+Y(2)+\cdots+Y(t)]=\frac{1}{t}\sum_{i=1}^{t}Y(i)$$

当得到第 t+1 期的实际值后，可计算出第 t+1 期的预测误差为：$e_{t+1}=Y(t+1)-\hat{Y}_{t+1}$。

简单平均法适合对较为平稳的时间序列进行预测，即当时间序列没有趋势时，该方法比较好。但如果时间序列有趋势或季节性时，该方法的预测不够准确。此外，简单平均法将远期的数值和近期的数值看做对未来的预测同等重要。但从预测角度来看，近期的数值通常要比远期的数值对预测结果影响更大。

2. 移动平均法

通过对时间序列逐期递移求得平均数作为预测值，这种预测方法称为移动平均法。它是对简单平均法的改进，可分为简单移动平均法和加权移动平均法。

（1）简单移动平均法

简单移动平均法是把第 t+1 期之前最近的 T 期数据加以平均作为第 t+1 期的预测值，这里 $T<t+1$：

$$\hat{Y}_{t+1}=\frac{1}{T}[Y(t)+Y(t-1)+\cdots+Y(t-T+1)]=\frac{1}{T}\sum_{i=1}^{T}Y(t-i+1)$$

T 称为移动周期。

同理，第 t+2 期的预测值为

$$\hat{Y}_{t+2}=\frac{1}{T}[Y(t+1)+Y(t)+\cdots+Y(t-T+2)]=\frac{1}{T}\sum_{i=1}^{T}Y(t-i+2)$$

移动平均法只使用最近 T 期的数据，所以在应用时关键是确定合理的移动周期 T，对于同一个时间序列，采用不同的移动周期，预测的准确性是不同的。移动周期 T 的选择可用 MSE 检验法来确定。首先计算在各种不同周期 T 的情况下预测的均方误差，它是预测误差平方和的平均数：

$$\mathrm{MSE}_{(T)}=\frac{\sum_{i=T+1}^{n}[Y(i)-\hat{Y}_i]^2}{n-T}$$

其中，n 为序列长度，即总的期数或样本数量；$Y(i)$是第 i 期的观察值；$\hat{Y}_i$ 是第 i 期

的预测值。

然后进行比较，选择使得 MSE 最小的周期 T 为移动平均法的周期。

下面是一个对消费价格指数序列进行预测的例子（如表 9-1 所示）。分别采用移动周期为 3 年、5 年对居民消费价格指数进行了预测，通过计算两种移动周期下的均方误差，可以发现，就本序列而言，选择 5 年移动周期进行预测使得 MSE 更小一些（$MSE_{(3)}=89.55$，$MSE_{(5)}=87.36$），预测更精确。

表 9-1　简单移动平均法预测消费价格指数

年度	消费价格指数	3 年移动平均值 $\hat{Y}_{t+1}$（k=3）	预测误差 e_{t+1}	误差平方 e_{t+1}^2	5 年移动平均值 $\hat{Y}_{t+1}$（k=5）	预测误差 e_{t+1}	误差平方 e_{t+1}^2
1986	106.5						
1987	107.3						
1988	118.8						
1989	118	110.87	7.13	50.88			
1990	103.1	114.70	–11.60	134.56			
1991	103.4	113.30	–9.90	98.01	110.74	–7.34	53.88
1992	106.4	108.17	–1.77	3.12	110.12	–3.72	13.84
1993	114.7	104.30	10.40	108.16	109.94	4.76	22.66
1994	124.1	108.17	15.93	253.87	109.12	14.98	224.40
1995	117.1	115.07	2.03	4.13	110.34	6.76	45.70
1996	108.3	118.63	–10.33	106.78	113.14	–4.84	23.43
1997	102.8	116.50	–13.70	187.69	114.12	–11.32	128.14
1998	99.2	109.40	–10.20	104.04	113.40	–14.20	201.64
1999	98.6	103.43	–4.83	23.36	110.30	–11.70	136.89
2000	100.4	100.20	0.20	0.04	105.20	–4.80	23.04
MSE	—	—	—	89.55	—	—	87.36

（2）加权移动平均法

简单移动平均法在预测时，将移动周期内的每个观察值都视为同等重要。但实际上近期的观察值和远期的观察值对预测的重要性是不同的。加权移动平均法就是在预测时，对近期的观察值和远期的观察值赋予不同的权值，再进行预测。通常，根据观察值时期的由近到远，相应的权值也应由大变小。所选择的各期的权值之和必须等于 1。

因此，第 t+1 期的预测值为

$$\hat{Y}_{t+1}=\alpha_1 Y(t-T+1)+\alpha_2 Y(t-T+2)+\cdots+\alpha_T Y(t)$$

其中，$\alpha_1+\alpha_2+\cdots+\alpha_T=1$。

在实际问题中，各期权值究竟应选择多大，是一个较难确定的问题，一般也是采用均方误差进行检验。检验方法是选择几组不同的权值，比较其预测结果，然后选择均方误差最小的一组权值。但在实际应用中，由于有更好的指数平滑预测法，加权移动平均法较少被采用。但加权移动平均法是指数平滑法的基础。

9.2 指数平滑法

9.2.1 指数平滑法概述

移动平均法的预测值实质上是以前观察值的加权和，且对不同时期的数据给予相同的加权。指数平滑法则对移动平均法进行了改进和发展。

指数平滑法是布朗（Robert G.Brown）所提出的，布朗认为时间序列的态势具有稳定性或规则性，所以时间序列可被合理地顺势推延；他认为最近的过去态势，在某种程度上会持续到最近的未来，所以将较大的权值放在最近的数据样本上。

指数平滑法是生产预测常用的一种方法，也用于中短期经济发展趋势预测，在所有预测方法中，指数平滑法是用得最多的一种。

指数平滑法是在移动平均法基础上发展起来的一种时间序列分析预测法，它是通过计算指数平滑值，配合一定的时间序列预测模型对现象的未来进行预测。其原理是任一期的指数平滑值都是本期实际观察值与前一期指数平滑值的加权平均。

根据平滑次数不同，指数平滑法分为：一次指数平滑法、二次指数平滑法和三次指数平滑法等。但它们的基本思想都是：预测值是以前观察值的加权和，且对不同的数据给予不同的权值，新数据给较大的权值，旧数据给较小的权值。

9.2.2 指数平滑模型

1. 简单指数平滑模型

简单指数平滑法（Simple Exponential Smoothing）是简单移动平均法的变形，也称为一次指数平滑法。当时间序列没有明显的趋势和季节性时（如图 9.1 所示），可以使用此方法。

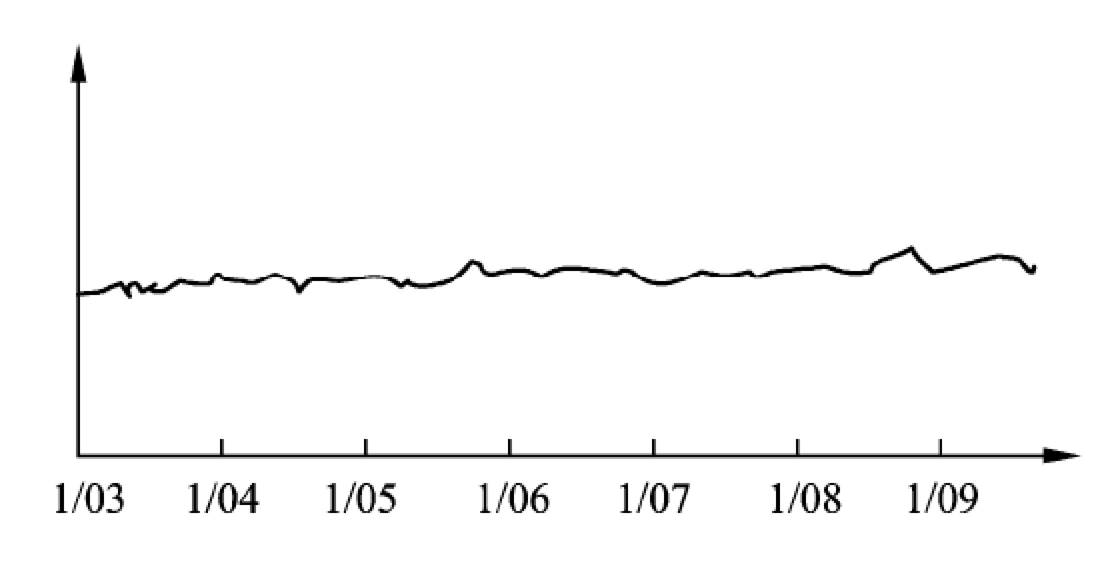


图 9.1　没有明显的趋势或季节性的时间序列

其模型可以用如下公式来描述：

$$L(t) = \alpha Y(t) + (1-\alpha)L(t-1)$$

其中，$L(t)$称为第 t 期的一次指数平滑值，$L(t-1)$ 为第$(t-1)$期的一次指数平滑值，α 称为“平滑系数”，其取值范围为[0,1]，$Y(t)$是第 t 期的实际观察值。

当根据已有的 t 期数据来预测第 t 期之后的第 k 期数据时，将预测值记为 $\hat{Y}_t(k)$ 。例

如当 k=1 时，即要预测第 t+1 期的值 $\hat{Y}_{t+1}$；当 k=2 时，即要预测第 t+2 期的值 $\hat{Y}_{t+2}$。

简单指数平滑法直接将 t 期的一次指数平滑值 $L(t)$作为第 t 期之后的预测值 $\hat{Y}_t(k)$：

$$\hat{Y}_t(k) = L(t) = \alpha Y(t) + (1-\alpha)L(t-1)$$

从式中可以看出，简单指数平滑法对第 $t+k$ 期（k=1,2,⋯）的预测结果，是第 t 期的观察值与第 t–1 期的预测值的加权平均值，权值分别为 α 和 $1-\alpha$。

由于 $L(t-1) = \alpha Y(t-1) + (1-\alpha)L(t-2)$，所以：

$$\begin{aligned}\hat{Y}_t(k) &= \alpha Y(t) + (1-\alpha)[\alpha Y(t-1) + (1-\alpha)L(t-2)] \\ &= \alpha Y(t) + (1-\alpha)\alpha Y(t-1) + (1-\alpha)^2 L(t-2) \\ &= \alpha[Y(t) + (1-\alpha)Y(t-1) + (1-\alpha)^2 Y(t-2) + (1-\alpha)^3 Y(t-3) + \cdots] + (1-\alpha)^t L(0)\end{aligned}$$

显然，随着时间逐渐向远推移，各期观察值对预测值的影响权值呈指数规律递减，这也正是该方法被称为“指数”平滑法的原因。

从简单指数平滑模型的公式可以看出，应用该模型需要首先确定两个参数，即平滑系数 α 和一次平滑指数的初始值 $L(0)$。

当 α 值较大时，对时间序列的修匀程度较小，平滑后的序列能够较快地反映出原序列的变化情况。因此适用于变化较大的时间序列；反之，α 值较小时，适用于变化较小的时间序列。在实际应用中，α 值需要通过试验比较才能确定，选优的标准是使预测误差最小。

需要指定 $L(0)$，是因为在计算 $L(1) = \alpha Y(1) + (1-\alpha)L(0)$ 时，$L(0)$无法根据观察值来确定（$Y(1)$已经是最远的观察值了）。通常，$L(0)$直接用 $Y(1)$来代替，即 $L(0)= Y(1)$。或者，用最远的 3 期观察值的平均值作为初始值：$L(0) = \dfrac{Y(1)+Y(2)+Y(3)}{3}$。

2．布朗单一参数指数平滑模型

布朗单一参数指数平滑法（Brown's Single Parameter Exponential Smoothing）属于二次指数平滑法。

二次指数平滑法（也称线性指数平滑法）是对一次指数平滑值再进行一次平滑。一次平滑法是直接利用平滑值作为预测值，而二次指数平滑则是利用平滑值对时间序列的线性趋势进行修正，更能消除原序列的不规则变动和周期性变动，使序列的长期趋势更加明显。二次指数平滑法包括布朗单一参数线性指数平滑、霍特双参数指数平滑等。

布朗单一参数指数平滑模型适合于其中有线性趋势但没有季节性的序列（如图 9.2 所示）。

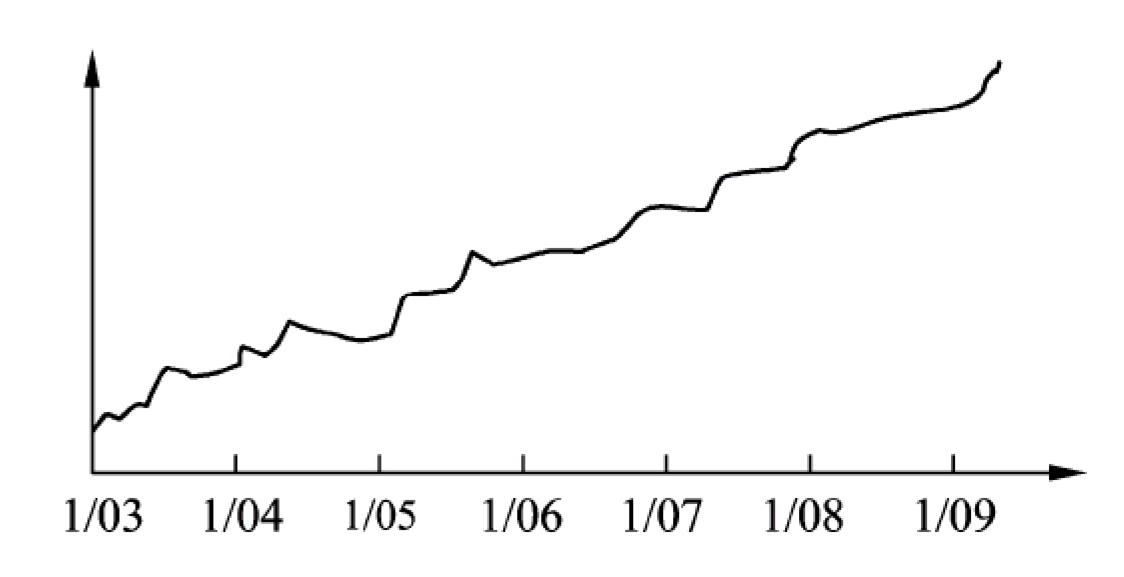


图 9.2　具有线性趋势无季节性的时间序列

其模型可以用如下公式来描述：

$$L(t)=\alpha Y(t)+(1-\alpha)L(t-1)$$
$$T(t)=\alpha(L(t)-L(t-1))+(1-\alpha)T(t-1)$$
$$\hat{Y}_t(k)=L(t)+((k-1)+\alpha^{-1})T(t)$$

下面是一个应用布朗单一参数指数平滑法进行预测的例子：

某地 1983—1993 年财政收入的资料如下，试用布朗单一参数指数平滑法预测 1996 年的财政收入。计算过程如表 9-2 所示。

表 9-2　布朗单一参数指数平滑计算过程

年份	t	财政收入/亿元	$L(t)$ $\alpha Y(t)+(1-\alpha)L(t-1)$ α=0.9，初始值 $L(0)$设为 23	$T(t)$ $\alpha(L(t)-L(t-1))+(1-\alpha)T(t-1)$ α=0.9，初始值 $T(0)$为 0
1983	1	29	28.40	4.86
1984	2	36	35.24	6.64
1985	3	40	39.52	4.52
1986	4	48	47.15	7.32
1987	5	54	53.32	6.28
1988	6	62	61.13	7.66
1989	7	70	69.11	7.95
1990	8	76	75.31	6.37
1991	9	85	84.03	8.49
1992	10	94	93.00	8.92
1993	11	103	102.00	8.99

由于要预测 1996 年的收入，也就是在 t=11（1993 年）处，向前预测 k=3 期的预测值。

所以，$\hat{Y}_{11}(3)=L(11)+\left((3-1)+\dfrac{1}{0.9}\right)T(11)=102+\left(2+\dfrac{1}{0.9}\right)\times 8.99=130$。

3．霍特双参数指数平滑模型

此模型适合于其中有线性趋势但没有季节性的序列。它比布朗更加常用，但在计算大型序列的估计值时会花费更多的时间。

霍特双参数指数平滑法（Holt’s Two-Paramete Exponential Smoothing）与布朗单一参数线性指数平滑法在原理上基本相似，但霍特法不是直接用二次指数平滑值进行计算，而是分别对原序列和序列的趋势进行平滑。它使用两个参数，分别为 α 和 γ，二者的取值范围均在 0~1 之间。

霍特双参数指数平滑可用如下公式来描述：

$$L(t)=\alpha Y(t)+(1-\alpha)(L(t-1)+T(t-1))$$
$$T(t)=\gamma(L(t)-L(t-1))+(1-\gamma)T(t-1)$$
$$\hat{Y}_t(k)=L(t)+kT(t)$$

这里，$L(t)$称为数据平滑值，$T(t)$为趋势平滑值。在简单指数平滑法中，第 t+1 期的

估计值是第 t 期的观察值与第 t 期估计值的加权平均。如果序列有趋势，$L(t)$和 $L(t+1)$之间就存在趋势差 $T(t)$。当序列呈上升趋势时，$L(t+1)$会低于实际值，当序列呈下降趋势时，$L(t+1)$会高于实际值。因此，霍特的方法是一种改进，在估计 $L(t)$时，给 $L(t–1)$加上一个趋势增量 $T(t–1)$，解决了估计值的时间滞后问题。这里的这个趋势增量也是一个指数平滑估计值：第 t 期的趋势估计值 $T(t)$是第 $t–1$ 期的趋势估计值 $T(t–1)$与这两期估计值之差 $L(t)–L(t–1)$的加权平均值，也就是用第 t 期与第 $t–1$ 期趋势平滑值之差来修正第 $t–1$ 期的趋势值。

在用霍特法进行预测时，平滑常数 α 和 γ 也需要选择几组不同的值来进行分析和比较，以预测误差最小的一组来作为平滑常数。

4. 阻尼趋势指数平滑

霍特模型在考查时间序列中可能存在的固有趋势（递增或者递减）时，它假设这个趋势在时间上是永久持续的。也就是说，无论时间发展多久，每一期与其前一期相比都有一个相对稳定的趋势增量 $T(t–1)$。然而对现实中的许多时间序列来说，这个假设往往是不切实际的。时间序列随着时间的推移，趋势增量 $T(t–1)$往往是递减的。例如一个百米运动员的训练成绩组成的时间序列，在刚开始进行训练时，成绩的增幅往往较大，但随着时间的推移，成绩的提高幅度会越来越小。所以这个假设可能会导致对未来超出实际的预测。

阻尼趋势指数平滑（Damped-Trend Exponential Smoothing）是对霍特模型的调整，用于对具有逐渐衰退的线性趋势但没有季节性的序列（如图 9.3 所示）进行预测。它将一个逐渐衰退的趋势考虑进来，这个衰退趋势会影响到后续期间的预测值。因此除了霍特模型中的两个参数外，它还包括第 3 个参数ϕ，一个介于 0～1 之间的数，用来表示趋势中的衰退比例。

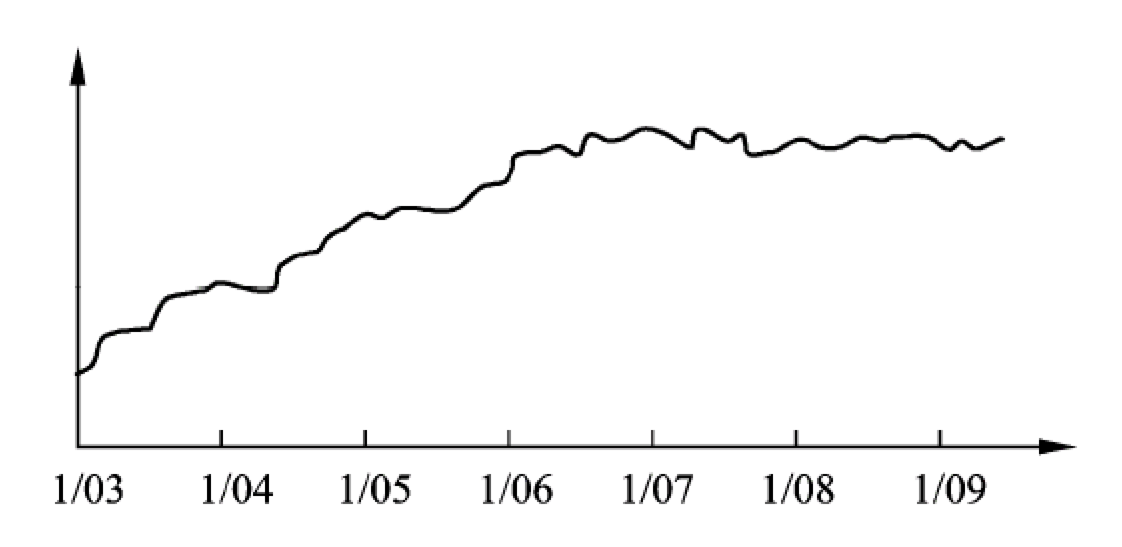


图 9.3 具有阻尼趋势无季节性的时间序列

阻尼趋势指数平滑可用如下公式来描述：

$$L(t) = \alpha Y(t) + (1-\alpha)(L(t-1) + \phi T(t-1))$$
$$T(t) = \gamma(L(t) - L(t-1)) + (1-\gamma)\phi T(t-1)$$
$$\hat{Y}_t(k) = L(t) + \sum_{i=1}^{k} \phi^i T(t)$$

5. 简单季节指数平滑

季节性变动是客观事物常见的一种变化规则，例如瓜果、服装的销量，会随着季节

的不同出现周期性的变动，铁路、航空的客运量会随着节假日出现周期性变动。所以，季节性时间序列是一种非常常见的时间序列。季节变动是有规律的，其表现形式为以年为周期，逐年的同月或同季有相同的变化方向和大致相同的变动幅度。

有些简单的季节性时间序列仅仅包含季节效应，且每年的季节效应比较稳定，不随时间发生显著的变化（如图 9.4 所示）。

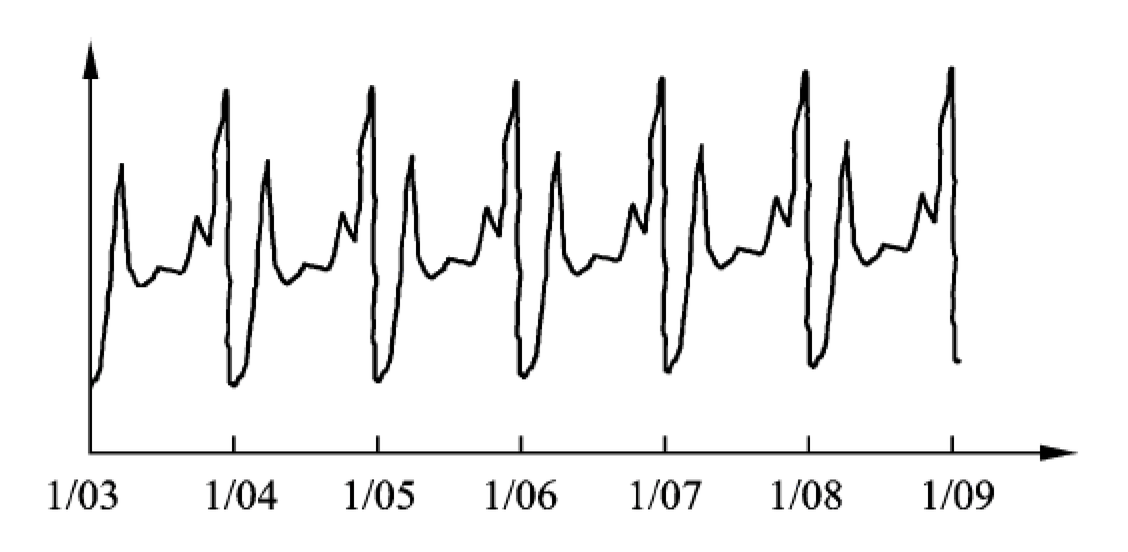


图 9.4　无趋势且季节效应不随时间变化的序列

有些比较复杂的季节性时间序列除了稳定的季节效应之外，还包含逐渐递增或递减的趋势特性（如图 9.5 所示）。

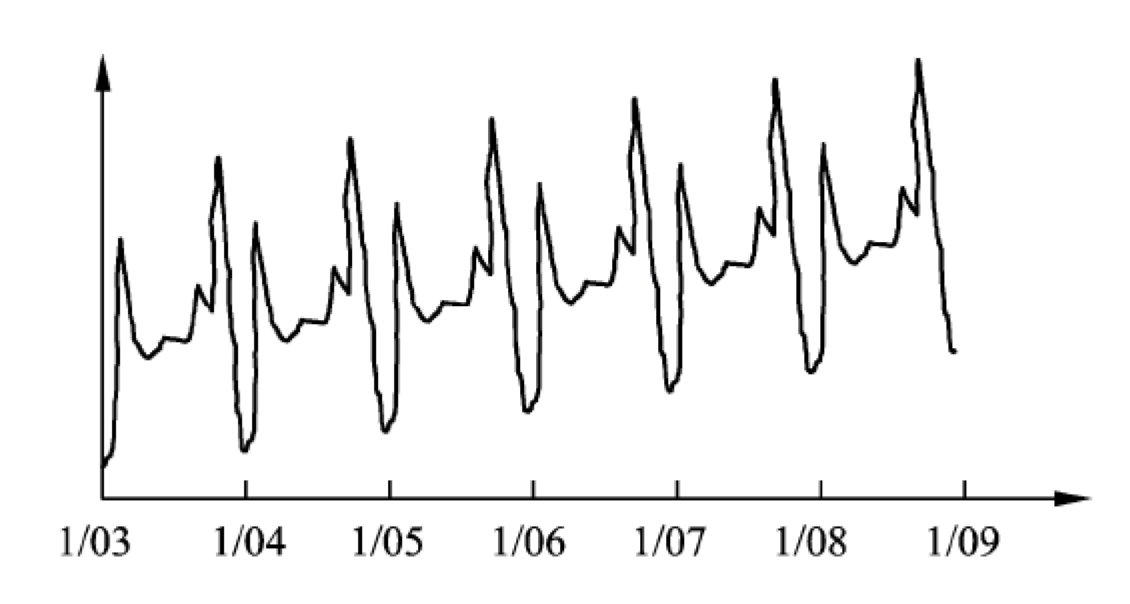


图 9.5　线性趋势且季节效应不随时间变化的序列

更为复杂的季节性时间序列则包含了季节性、趋势性、随机变动等多种因素，如图 9.6 所示的时间序列，就包含了递增的趋势和季节效应，且季节效应并不稳定，随时间在逐渐变化。

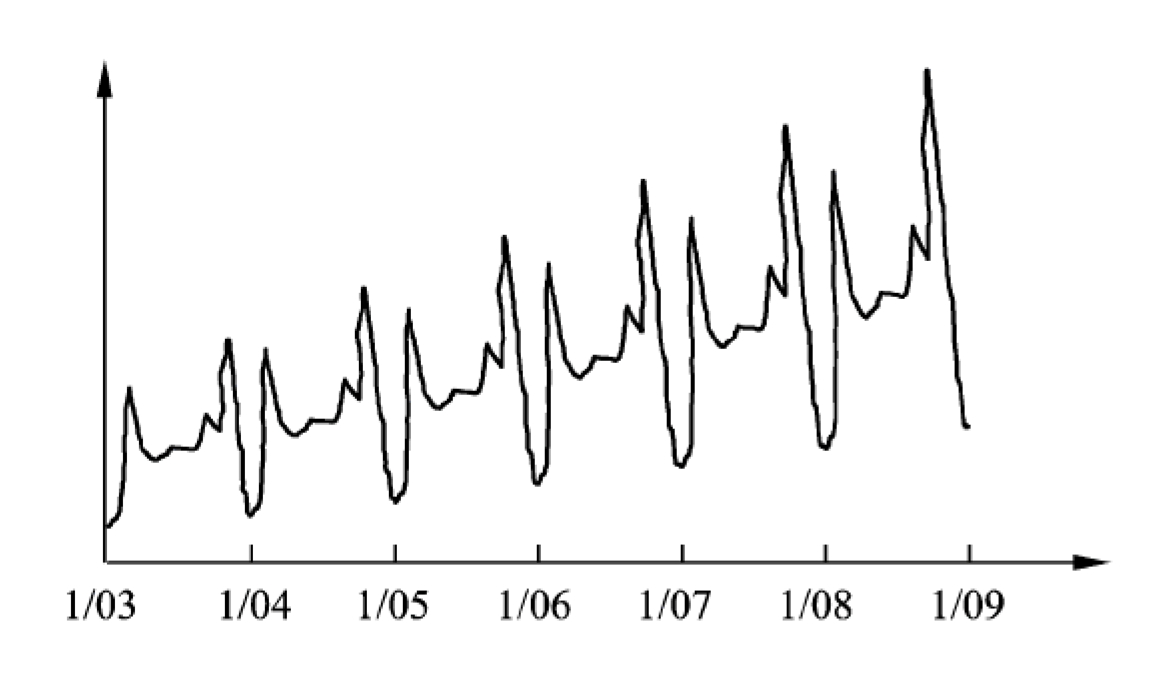


图 9.6　线性趋势且季节效应随时间变化的序列

简单季节指数平滑（Simple Seasonal Exponential Smoothing）模型适合于类似图 9.4

所示的没有趋势且季节效应不随时间变化的时间序列。其模型可用如下公式来描述：

$$L(t) = \alpha(Y(t) - S(t-s)) + (1-\alpha)L(t-1)$$

$$S(t) = \delta(Y(t) - L(t)) + (1-\delta)S(t-s)$$

$$\hat{Y}_t(k) = L(t) + S(t+k-s)$$

这里，$L(t)$为数据平滑值，$S(t)$为季节平滑值，s 是周期长度（也就是一年当中包含的样本数量），δ 为季节平滑参数，一个介于 0～1 之间的数。

6．温特加法指数平滑模型

温特加法指数平滑模型（Winters' Additive Exponential Smoothing）适合于类似图 9.5 所示的具有线性趋势且季节效应不随时间变化的序列。其模型可用如下公式来描述：

$$L(t) = \alpha(Y(t) - S(t-s)) + (1-\alpha)(L(t-1) + T(t-1))$$

$$T(t) = \gamma(L(t) - L(t-1)) + (1-\gamma)T(t-1)$$

$$S(t) = \delta(Y(t) - L(t)) + (1-\delta)S(t-s)$$

$$\hat{Y}_t(k) = L(t) + kT(t) + S(t+k-s)$$

其中，$L(t)$为数据平滑值，$T(t)$为趋势平滑值，$S(t)$为季节平滑值，α 、γ 和 δ 是 3 个平滑参数。

7．温特乘法指数平滑模型

温特乘法指数平滑模型（Winters' Multiplicative Exponential Smoothing）适合于类似图 9.6 所示的具有线性趋势且季节效应随序列的大小变化的序列。其模型可用如下公式来描述：

$$L(t) = \frac{\alpha Y(t)}{S(t-s)} + (1-\alpha)(L(t-1) + T(t-1))$$

$$T(t) = \gamma(L(t) - L(t-1)) + (1-\gamma)T(t-1)$$

$$S(t) = \frac{\delta Y(t)}{L(t)} + (1-\delta)S(t-s)$$

$$\hat{Y}_t(k) = (L(t) + kT(t))S(t+k-s)$$

其中，$L(t)$为数据平滑值，$T(t)$为趋势平滑值，$S(t)$为季节平滑值，α、γ 和 δ 是 3 个平滑参数。

可以看出，不同的模型各有不同的适用场合。因此在实际应用当中，首先要分析时间序列中的趋势性、季节性等特点，然后选择合适的模型。

9.3 ARIMA 模型

ARIMA 模型，即自回归求和移动平均模型（Autoregressive Integrated Moving Average Model），是由博克思（Box）和詹金斯（Jenkins）于 20 世纪 70 年代初提出的一个著名时间序列预测方法，所以又称为 box-jenkins 模型、博克思-詹金斯法。

ARIMA 模型的基本思想是：将预测对象随时间推移而形成的数据序列视为一个随机序列，用一定的数学模型来近似描述这个序列。这个模型一旦被识别后就可以从时间序列的过去值及现在值来预测未来值。

在讨论 ARIMA 模型之前，应先了解 ARMA 模型。

9.3.1 ARMA 模型

ARMA 模型，即自回归移动平均模型（Autoregressive Moving Average Model），是目前最常用的拟合平稳随机序列的模型。

所谓平稳随机序列，直观地说，其折线图没有明显的上升或下降的趋势，统计特性不随时间的推移而变化。另外，ARMA 模型预测的平稳随机序列还必须是零均值的，即观察值的大小围绕着 0 上下随机波动。

然而，大量的社会经济现象都是随着时间的推移表现出某种上升或下降的趋势，构成非零均值的非平稳的时间序列。因此，在应用 ARMA 模型之前，往往需要先对时间序列进行零均值化和差分平稳化处理。

所谓零均值化处理，是指对均值不为零的时间序列中的每一个项数值都减去该序列的平均数，构成一个均值为零的新的时间序列，即：

$$Y_t' = Y_t - \overline{Y}$$

其中，$\overline{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t$，是时间序列的平均值，$n$ 是该序列的数据的个数。

所谓差分平稳化处理，是指对均值为零的非平稳的时间序列进行差分处理，使之成为平稳时间序列，详细介绍请见第 9.3.2 节。

1. ARMA 模型概述

ARMA 模型可细分为自回归模型（Autoregressive Model，AR 模型）、移动平均模型（Moving Average Model，MA 模型）和自回归移动平均模型（Autoregressive Moving Average Model，ARMA 模型）。

（1）AR 模型

AR 模型与线性回归模型比较相似。在线性回归分析中，用下式来表示变量之间存在的某种线性的相关关系：

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

这里，$x_1, x_2, \cdots, x_p$ 是自变量；y 是因变量；ε 称为误差项，是一个随机变量，反映了其他随机因素对 y 的影响。

AR 模型认为，时间序列 Y_t 的变化受到自身变化的影响（即受到其之前若干期观察值的影响），因此同样可以用回归模型来表达时间序列的这种不同时期的相关性：

$$Y_t = \varphi_0 + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + e_t$$

其中，Y_t 是时间序列在第 t 期的观察值，Y_{t-1} 是时间序列在第 t–1 期的观察值，类似地，Y_{t-p} 是时间序列在第 t–p 期的观察值，p 是 AR 模型的阶数，表示 Y_t 仅与其之前的 p

期序列值相关，而与其间隔超过 p 期的序列值不再相关。e_t 是误差，表示不能用模型说明的随机因素。

当 $\varphi_0=0$ 时，Y_t 是一个零均值的时间序列，于是：

$$Y_t=\varphi_1 Y_{t-1}+\varphi_2 Y_{t-2}+\cdots+\varphi_p Y_{t-p}+e_t$$

就是 AR 模型的表达公式，记为 AR(p)。

（2）MA 模型

MA 模型的基本思想是，时间序列模型可以根据平均前期预测误差的原则来建立，在前期预测值之上加上预测误差便可得到现在的预测值。于是通过递推，可得到 MA(q) 模型为：

$$Y_t=e_t-\theta_1 e_{t-1}-\theta_2 e_{t-2}-\cdots-\theta_q e_{t-q}$$

记为 MA(q)，其中，Y_t 是时间序列在第 t 期的观察值，q 是 MA 模型的阶数，e_t 是时间序列在第 t 期的误差，e_{t-1} 是时间序列在第 $t-1$ 期的误差，类似地，e_{t-q} 是时间序列在第 $t-q$ 期的误差。

（3）ARMA 模型

ARMA 模型是建立在 AR 模型和 MA 模型基础之上的。将 AR 模型和 MA 模型有效地组合起来，就构成了 ARMA 模型：

$$Y_t=\varphi_1 Y_{t-1}+\varphi_2 Y_{t-2}+\cdots+\varphi_p Y_{t-p}+e_t-\theta_1 e_{t-1}-\theta_2 e_{t-2}-\cdots-\theta_q e_{t-q}$$

记为 ARMA(p,q)。其中，p 称为自回归阶数，$\{\varphi_1,\varphi_2,\cdots,\varphi_p\}$ 为自回归系数；q 称为移动平均阶数，$\{\theta_1,\theta_2,\cdots,\theta_p\}$ 为移动平均系数。

显然，当 p=0 时，模型就是移动平均模型，记为 ARMA(0,q)；当 q=0 时，模型就是自回归模型，记为 ARMA(p,0)。

2. 自相关分析

在建立 ARMA 模型之前，必须首先确定模型的类型（即选定 AR 模型、MA 模型还是 ARMA 模型）以及模型的阶数（p 和 q 的值）。这些内容的确定是以时间序列的自相关分析为基础的。自相关分析就是对时间序列求其本期与不同滞后期的一系列自相关系数和偏自相关系数，并据此来识别时间序列的特性。

（1）自相关系数

自相关是时间序列 Y_1，Y_2，…，Y_t 各项之间的简单相关，它的含义与相关分析中变量之间的相关一样，只是因为所涉及的是同一序列自身，因而称做自相关，自相关系数记为 r_k，即 k 阶自相关系数，表示序列中任意相隔 k 期的两项之间的相关程度。例如，r_1 表示每相邻两项之间的相关程度；r_2 表示每隔一项的两个观察值的相关程度。自相关系数的计算公式为：

$$r_k=\frac{\sum_{t=k+1}^{n}(Y_t-\overline{Y})(Y_{t-k}-\overline{Y})}{\sum_{t=1}^{n}(Y_t-\overline{Y})^2}$$

式中，n 是时间序列中观察值的数量；$\overline{Y}=\frac{1}{n}\sum_{t=1}^{n}Y_t$ 是时间序列的平均值。显然，对于零均值的时间序列，其自相关系数为：

$$r_k=\frac{\sum_{t=k+1}^{n}Y_tY_{t-k}}{\sum_{t=1}^{n}Y_t^2}$$

自相关系数 r_k 的取值范围在–1～1 之间，$|r_k|$与 1 越接近，说明时间序列的自相关程度越高。自相关系数可以提供时间序列及其模式构成的重要信息。对于纯随机序列，即一个由随机数字构成的时间序列，其各阶的自相关系数将接近于零或等于零。而具有明显的上升或下降趋势的时间序列或具有强烈季节变动或循环变动性质的时间序列，将会有高度的自相关。

下面是一个计算自相关系数的例子。表 9-3 是一个时间序列 Y_t 的数据，试计算其自相关系数 r_1, r_2, r_3, r_4。

表 9-3　时间序列 Y_t 的观察值

t	1	2	3	4	5	6	7	8
Y_t	16	13	5	7	23	27	3	5

首先求出 $\overline{Y}$ =12.375，并将该时间序列零均值化。计算过程如表 9-4 所示。

表 9-4　自相关系数的计算过程

t	Y_t	Y_{t-1}	Y_{t-2}	Y_{t-3}	Y_{t-4}	$Y_t\times Y_{t-1}$	$Y_t\times Y_{t-2}$	$Y_t\times Y_{t-3}$	$Y_t\times Y_{t-4}$
1	3.625	—	—	—	—	—	—	—	—
2	0.625	3.625	—	—	—	2.266	—	—	—
3	–7.375	0.625	3.625	—	—	–4.609	–26.734	—	—
4	–5.375	–7.375	0.625	3.625	—	39.641	–3.359	–19.484	—
5	10.625	–5.375	–7.375	0.625	3.625	–57.109	–78.359	6.641	38.516
6	14.625	10.625	–5.375	–7.375	0.625	155.391	–78.609	–107.859	9.141
7	–9.375	14.625	10.625	–5.375	–7.375	–137.109	–99.609	50.391	69.141
8	–7.375	–9.375	14.625	10.625	–5.375	69.141	–107.859	–78.359	39.641
$\sum_{t=1}^{8}Y_t^2=565.875$						$\sum_{t=2}^{8}Y_tY_{t-1}$	$\sum_{t=3}^{8}Y_tY_{t-2}$	$\sum_{t=4}^{8}Y_tY_{t-3}$	$\sum_{t=5}^{8}Y_tY_{t-4}$
						67.609	–394.531	–148.672	156.438

所以：

$$r_1=\frac{\sum_{t=2}^{8}Y_tY_{t-1}}{\sum_{t=1}^{8}Y_t^2}=\frac{67.609}{565.875}=0.119 \qquad r_2=\frac{\sum_{t=3}^{8}Y_tY_{t-2}}{\sum_{t=1}^{8}Y_t^2}=\frac{-394.531}{565.875}=0.697$$

$$r_3=\frac{\sum_{t=4}^{8}Y_tY_{t-3}}{\sum_{t=1}^{8}Y_t^2}=\frac{-148.672}{565.875}=-0.263 \qquad r_4=\frac{\sum_{t=5}^{8}Y_tY_{t-4}}{\sum_{t=1}^{8}Y_t^2}=\frac{156.438}{565.875}=0.276$$

（2）偏自相关系数

在时间序列中，偏自相关是在给定了 Y_{t-1}，Y_{t-2}，…，Y_{t-k+1} 的条件下，Y_t 与 Y_{t-k} 之间的条件相关。它用以测度当其他滞后期 k=1，2，…，k–1 时间序列的作用能够已知的条件下，Y_t 与 Y_{t-k} 之间的相关程度。由于它需要考虑排除其他滞后期的效应，因而被称做偏自相关。其相关程度用偏自相关系数 ϕ_{kk} 来度量，$-1\leqslant\phi_{kk}\leqslant 1$，$\phi_{kk}$ 的计算公式为：

$$\phi_{kk}=\begin{cases} r_1 & k=1 \\ \dfrac{r_k-\sum_{i=1}^{k-1}(\phi_{k-1,i}\times r_{k-i})}{1-\sum_{i=1}^{k-1}(\phi_{k-1,i}\times r_i)} & k=2,3,\cdots \end{cases}$$

其中，$\phi_{k,i}=\phi_{k-1,i}-\phi_{kk}\times\phi_{k-1,k-i}$，$i=1,2,\cdots,k-1$。

在前面的例子中，已经求出了 r_1=0.119，r_2=0.697，r_3= –0.263，r_4=0.276。据此可以求出偏自相关系数 ϕ_{11}，ϕ_{22}，ϕ_{33}，ϕ_{44}。

$$\phi_{11}=r_1=0.119$$

$$\phi_{22}=\frac{r_2-\phi_{11}\times r_1}{1-\phi_{11}\times r_1}=\frac{r_2-r_1^2}{1-r_1^2}=-0.722$$

$$\phi_{21}=\phi_{11}-\phi_{22}\times\phi_{11}=0.206$$

$$\phi_{33}=\frac{r_3-(\phi_{21}\times r_2+\phi_{22}\times r_1)}{1-(\phi_{11}\times r_1+\phi_{22}\times r_2)}=-0.07$$

$$\phi_{31}=\phi_{21}-\phi_{33}\times\phi_{22}=0.156$$

$$\phi_{32}=\phi_{22}-\phi_{33}\times\phi_{21}=-0.708$$

$$\phi_{44}=\frac{r_4-(\phi_{31}\times r_3+\phi_{32}\times r_2+\phi_{33}\times r_1)}{1-(\phi_{11}\times r_1+\phi_{22}\times r_2+\phi_{33}\times r_3)}=-0.357$$

3．ARMA 模型的自相关和偏相关函数

平稳序列的自相关系数是关于时滞 k 的函数。因此，序列的自相关系数构成自相关函数，序列的偏自相关系数构成偏自相关函数。

（1）AR(p)模型的自相关函数和偏自相关函数

AR(p)模型为 $Y_t=\varphi_1Y_{t-1}+\varphi_2Y_{t-2}+\cdots+\varphi_pY_{t-p}+e_t$，其偏自相关系数满足：

$$\phi_{ki}=\begin{cases}\phi_i & 1\leqslant i\leqslant p\\ 0 & p+1\leqslant i\leqslant k\end{cases}$$

所以，AR(p)模型的 ϕ_{kk} 是 p 步截尾的，即：

$$\phi_{kk}=\begin{cases}\neq 0 & k\leqslant p\\ =0 & k>p\end{cases}$$

例如，AR(1)模型和 AR(2)模型的偏自相关函数如图 9.7 所示。

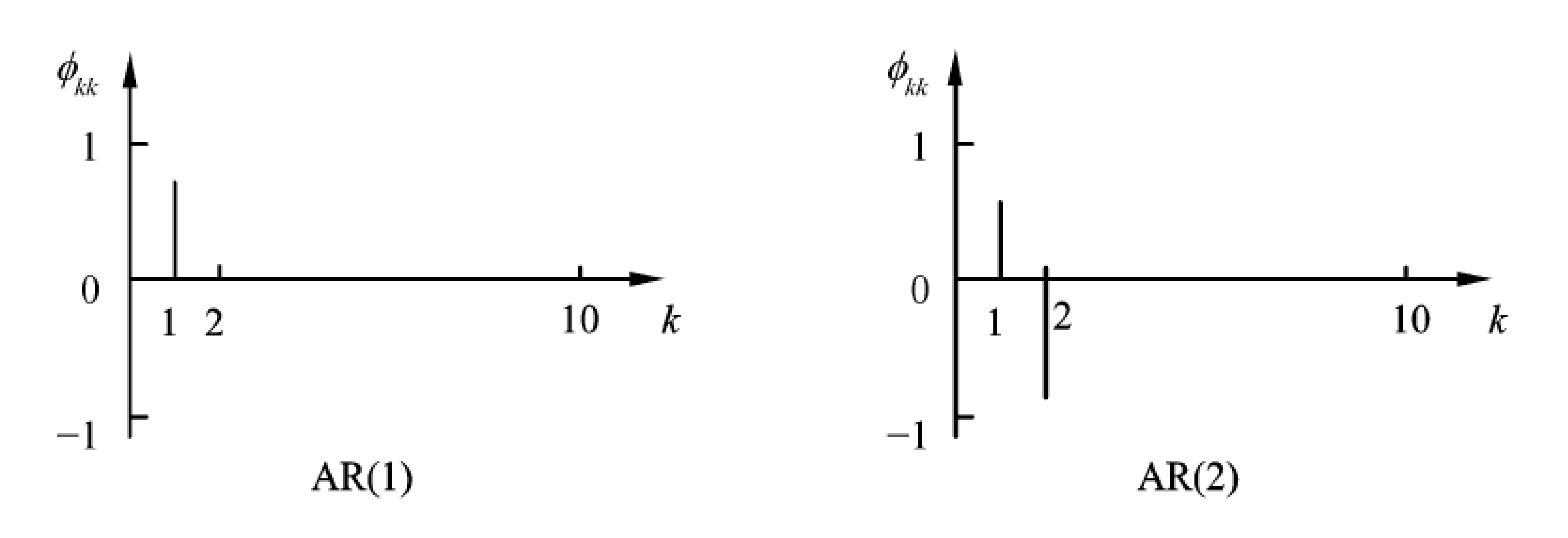


图 9.7　AR(*p*)序列偏自相关函数的 *p* 步截尾性

AR(p)模型偏自相关函数的 p 步截尾性对于识别 AR(p)模型具有十分重要的意义。

AR(p)模型的自相关函数 r_k 则随时滞 k 的增加，呈指数衰减或正弦波衰减并趋于零，这种性质称为自相关函数的拖尾性。AR(p)模型的自相关函数如图 9.8 所示。

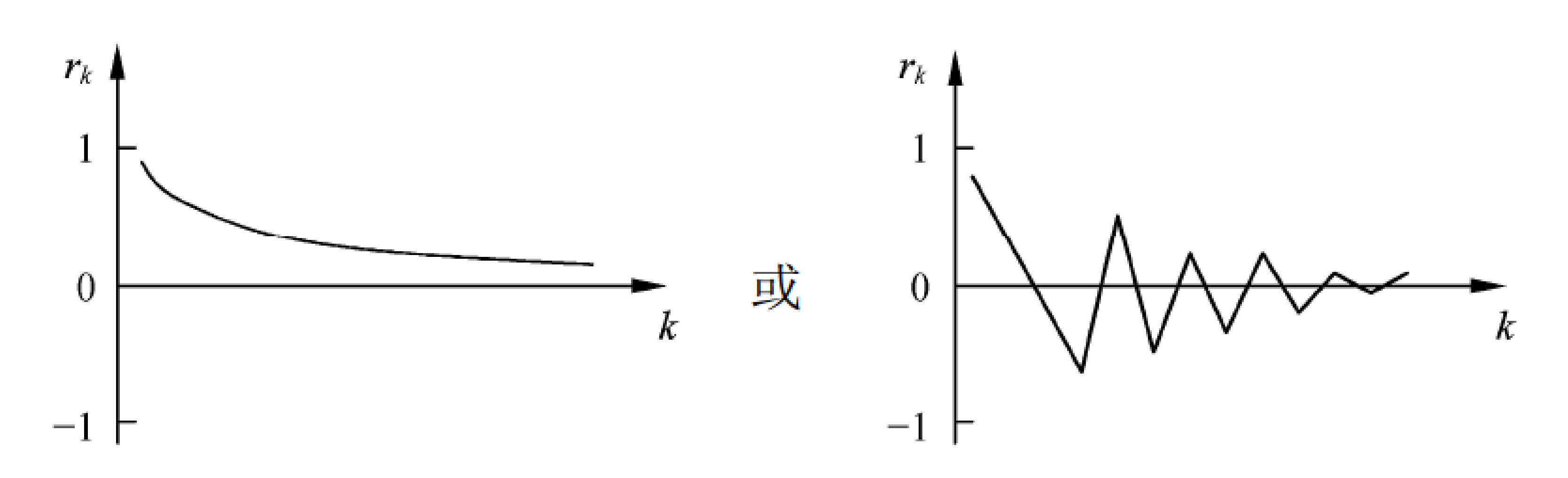


图 9.8　AR(*p*)序列自相关函数的拖尾性

（2）MA(q)模型的自相关函数和偏自相关函数

MA(q)模型为 $Y_t=e_t-\theta_1 e_{t-1}-\theta_2 e_{t-2}-\cdots-\theta_q e_{t-q}$，其自相关函数为：

$$r_k=\begin{cases}\dfrac{-\theta_k+\theta_1\theta_{k+1}+\cdots+\theta_{q-k}\theta_q}{1+\theta_1^2+\theta_2^2+\cdots+\theta_q^2} & 1\leqslant k\leqslant q\\ 0 & k>q\end{cases}$$

上式说明，对于 MA(q)序列，当 Y_t 和 Y_s 之间的间隔 k=t–s 满足条件 k>q 时，Y_t 和 Y_s 之间不相关，也就是说当 k>q 后，MA(q)模型的自相关函数 r_k 全部等于 0。但由于 r_q 不等于 0，所以 MA(q)模型的自相关函数具有 q 步截尾性。这一性质被用来识别 MA 模型及确定模型的阶数 q。

MA(1)模型和 MA(2)模型的自相关函数如图 9.9 所示。

MA(q)模型的偏自相关函数与自相关函数不同，不能在某步之后截尾，而是随时滞 k 的增加，呈指数衰减或衰减的正弦波，趋向于 0，即表现出拖尾性。图 9.10 是 MA(q)模型的偏自相关函数。

（3）ARMA(p,q)模型的自相关函数和偏自相关函数

ARMA(p,q)模型包含了两个过程，即自回归过程和移动平均过程，因而其自相关与

偏自相关函数都较单纯的 AR(p)和 MA(q)模型更复杂，均表现出拖尾性。

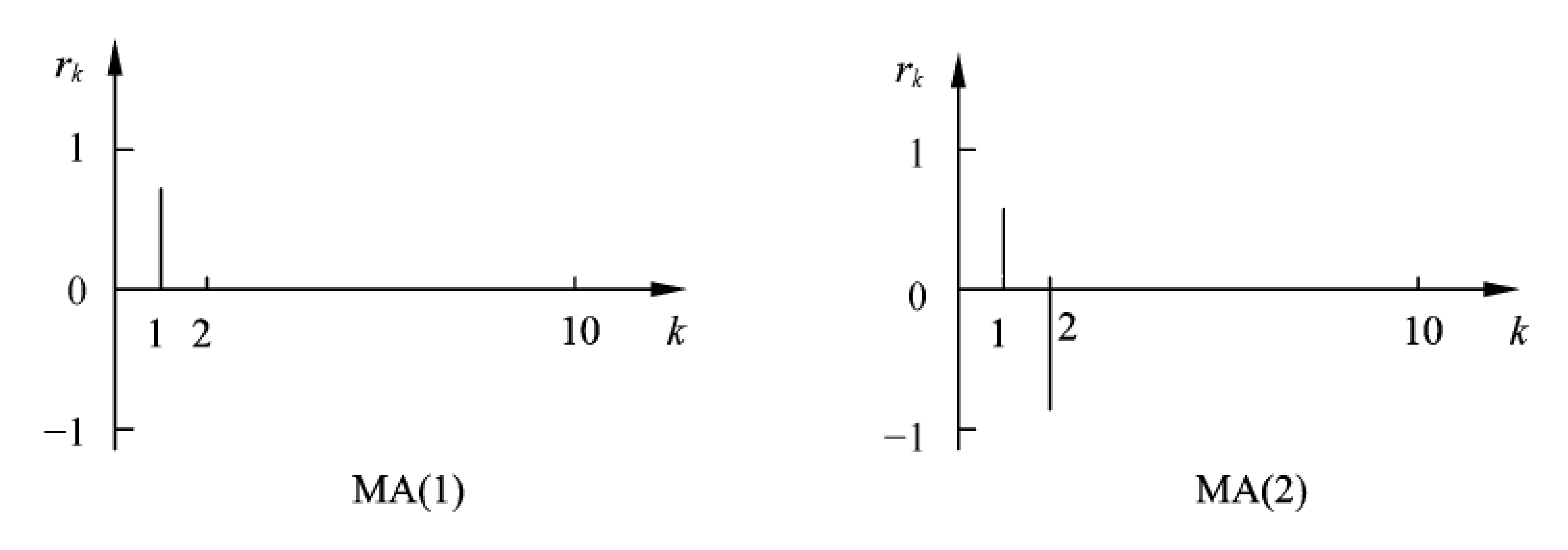


图 9.9　MA(q)序列自相关函数的 q 步截尾性

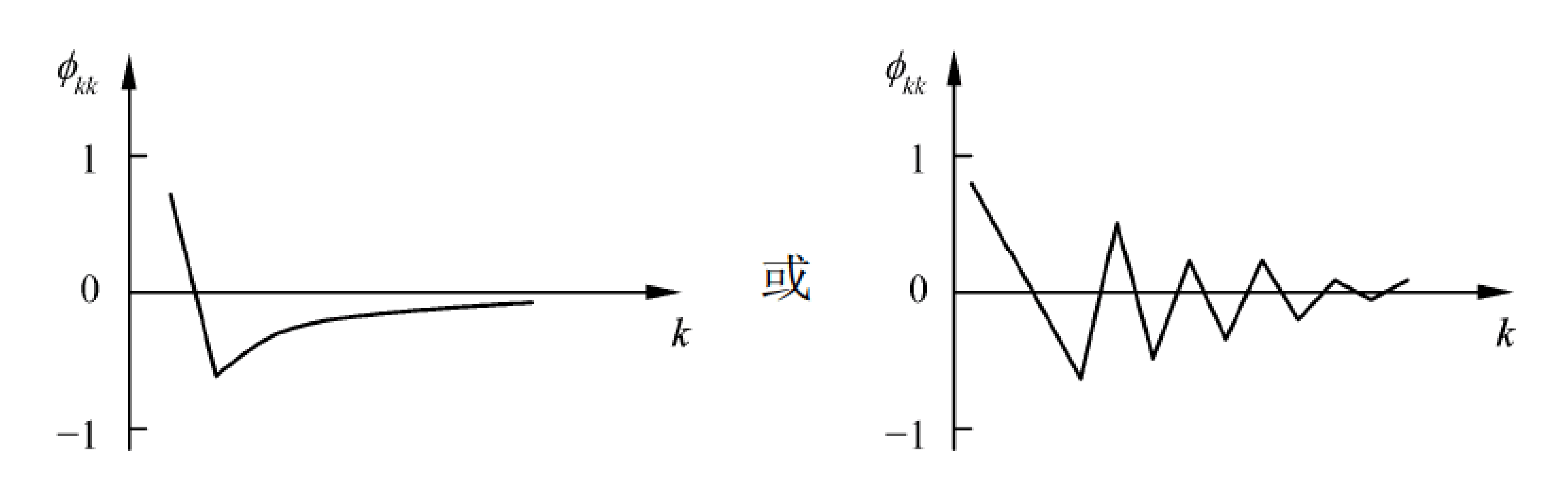


图 9.10　MA(q)序列偏自相关函数的拖尾性

综合考察 AR(p)模型、MA(q)模型和 ARMA(p,q)模型自相关系数和偏自相关系数的性质，可以总结出如下规律，如表 9-5 所示。

表 9-5　自相关系数和偏自相关系数的性质

模型	自相关函数	偏自相关函数
AR(p)	拖尾	p 步截尾
MA(q)	q 步截尾	拖尾
ARMA(p,q)	拖尾	拖尾

9.3.2　差分运算与 ARIMA 模型

ARMA 模型用于对平稳序列的拟合，但在实际问题中，许多序列并不是平稳序列，不能直接用 ARMA 模型去描述。因此在应用 ARMA 模型之前，必须先对非平稳的序列进行处理，得到平稳序列。非平稳序列可以通过差分运算成为平稳序列。

1．差分运算

（1）d 阶差分

相隔 1 期的两个序列值之间的减法运算称为 1 阶差分运算，记 ∇Y_t 为 Y_t 的 1 阶差分：

$$\nabla Y_t = Y_t - Y_{t-1}$$

对 1 阶差分后的序列再进行一次差分运算称为 2 阶差分，记为 $\nabla^2 Y_t$：

$$\nabla^2 Y_t = \nabla Y_t - \nabla Y_{t-1}$$

以此类推，对 d–1 阶差分后的序列再进行一次 1 阶差分运算称为 d 阶差分，记为 $\nabla^d Y_t$，Y_t 的 d 阶差分为：

$$\nabla^d Y_t = \nabla^{d-1} Y_t - \nabla^{d-1} Y_{t-1}$$

（2）k 步差分

相隔 k 期的两个序列值之间的减法运算称为 k 步差分运算。记 $\nabla_k Y_t$ 为 Y_t 的 k 步差分：

$$\nabla_k Y_t = Y_t - Y_{t-k}$$

k 步差分可用于处理序列的季节性。例如，如果序列的时间间隔为 1 个月，季节周期为 12 个月，那么可以对序列进行 12 步差分运算：

$$\nabla_{12} Y_t = Y_t - Y_{t-12}$$

（3）延迟算子

延迟算子用来表示相隔 1 期的序列值之间的关系。记 B 为延迟算子，那么：

$$Y_{t-1}=BY_t,\ \ Y_{t-2}=B^2Y_t,\ \ \cdots,\ \ Y_{t-d}=B^dY_t$$

因此，用延迟算子来表示 d 阶差分为：

$$\nabla^d Y_t = (1-B)^d Y_t$$

用延迟算子来表示 k 步差分为：

$$\nabla_k Y_t = Y_t - Y_{t-k} = (1-B^k)Y_t$$

通常，一个具有线性趋势的序列，可以通过 1 阶差分处理实现趋势平稳；如果序列具有曲线趋势，通过 2 阶或 3 阶差分后可以实现平稳化；如果序列蕴涵着固定周期，可以进行步长为周期长度的差分运算，消除周期信息的影响后得到平稳序列。

2. ARIMA 模型

（1）ARIMA(p,d,q)模型

设 Y_t 为一含有趋势性的非平稳序列，在经过 d 阶差分后得到平稳序列，记为 Z_t，

$$Z_t = \nabla^d Y_t = (1-B)^d Y_t \qquad t > d$$

且 Z_t 可用 ARMA(p,q)模型来表示，那么 Y_t 称为 ARMA 的 d 阶求和序列，并用 ARIMA(p,d,q)表示。其中，d 为求和阶数，p 和 q 分别为 Z_t 的自回归阶数和移动平均阶数。

设 Y_t 经过 1 阶差分（即 d=1）后得到平稳序列 Z_t，所以 $Z_t=(1-B)Y_t$。那么序列 Z_t 的 ARMA(1,1)模型可表示为

$$Z_t - \varphi_1 Z_{t-1} = e_t - \theta_1 e_{t-1}，即 (1-B)Y_t - \varphi_1(1-B)Y_{t-1} = e_t - \theta_1 B e_t$$

因此，序列 Y_t 的 ARIMA(1,1,1)模型可表示为：

$$(1-B)Y_t - \varphi_1(1-B)BY_t = e_t - \theta_1 B e_t，即 (1-B)(1-\varphi_1 B)Y_t = (1-\theta_1 B)e_t$$

一般地，ARIMA(p,d,q)模型可表示为

$$\varphi(B)(1-B)^d Y_t = \theta(B)e_t$$

（2）ARIMA(p,d,q)(P,D,Q)^s 模型

当序列中同时存在趋势性和季节性时，可以对序列进行 d 阶差分，消除趋势性，并

进行 D 阶 s 步差分（s 为周期），消除季节性。 所以，此类含有趋势性和季节性的非平稳时间序列需要用更复杂的模型来描述：$\text{ARIMA}(p,d,q)(P,D,Q)^s$。

一般地，$\text{ARIMA}(p,d,q)(P,D,Q)^s$ 可表示为

$$\varphi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D Y_t = \theta_q(B)\Theta_Q(B^s)e_t$$

其中，p 是非季节性自回归阶数，P 是季节性自回归阶数，q 是非季节性移动平均阶数，Q 是季节性移动平均阶数，且

$\varphi_p(B)=1-\varphi_1 B-\varphi_2 B^2-\cdots-\varphi_p B^p$ 是模型非季节的 $\text{AR}(p)$部分

$\Phi_P(B^s)=1-\Phi_1 B^s-\Phi_2 B^{2s}-\cdots-\Phi_P B^{Ps}$ 是模型季节的 $\text{AR}(P)$部分

$(1-B)^d$ 表示 d 阶逐期差分

$(1-B^s)^D$ 表示 D 阶 s 步季节差分

$\theta_q(B)=1-\theta_1 B-\theta_2 B^2-\cdots-\varphi_q B^q$ 是模型非季节的 $\text{MA}(q)$部分

$\Theta_Q(B^s)=1-\Theta_1 B^s-\Theta_2 B^{2s}-\cdots-\Phi_Q B^{Qs}$ 是模型季节的 $\text{MA}(Q)$部分

所以，时间序列模型 $\text{ARIMA}(p,d,q)(P,D,Q)^s$ 可以描述各种非平稳的时间序列，是时间序列最一般的表示形式，它包括了 $\text{AR}(p)$、$\text{MA}(q)$、$\text{ARMA}(p,q)$、$\text{ARIMA}(p,d,q)$、$\text{ARIMA}(P,D,Q)^s$ 以及各种组合模型。

9.3.3 ARIMA 建模过程

使用 ARIMA 模型对观测序列建模，主要包括平稳性检验、确定模型、参数估计、模型检验等过程。ARIMA 建模的基本流程如图 9.11 所示。

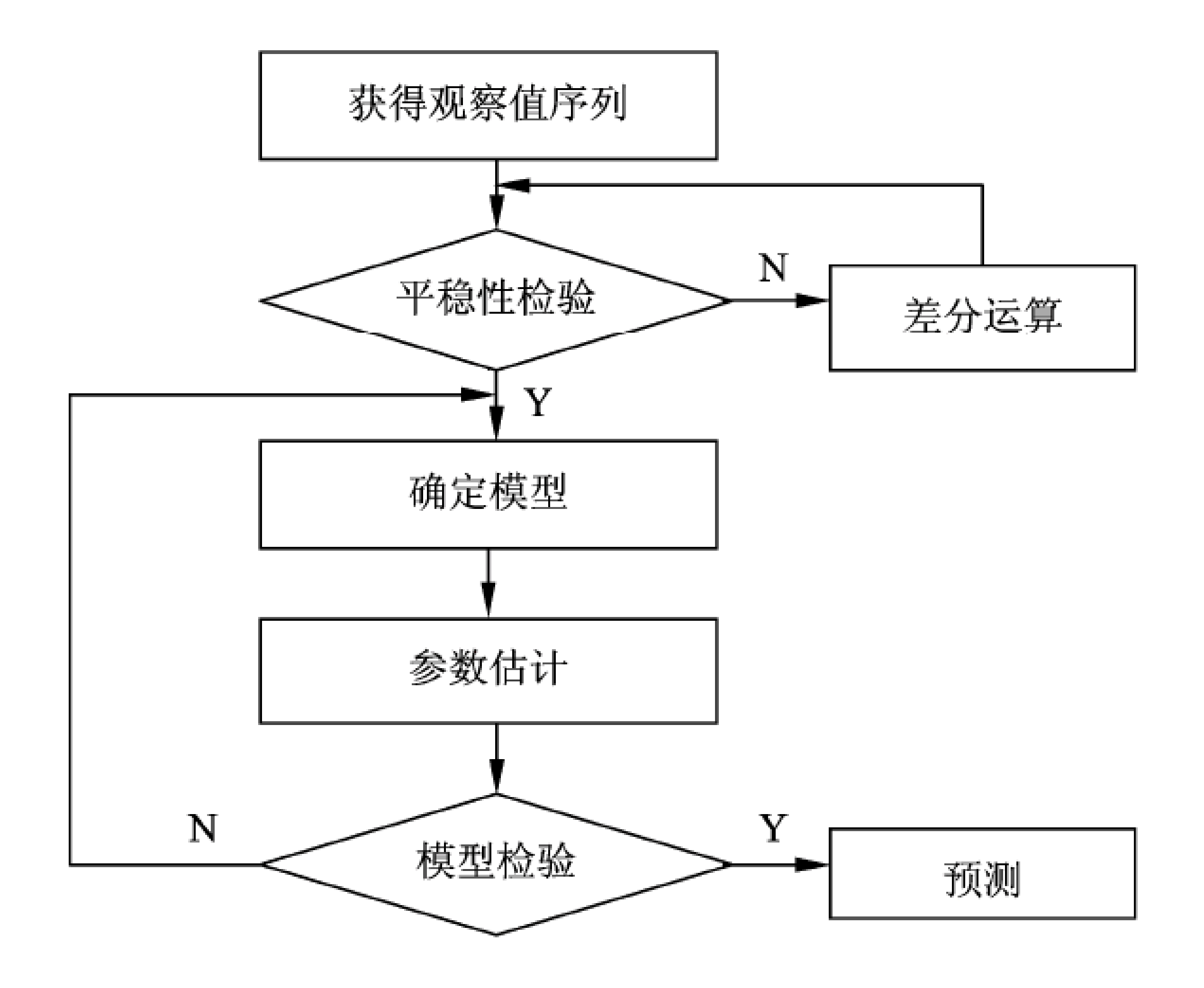


图 9.11 ARIMA 建模的基本流程

1. 平稳性检验

现实中的时间序列大多都不是平稳序列，只有将非平稳序列进行差分运算并通过平

稳性检验后，才可以使用 ARMA 模型来描述。也就是说，在确定 $ARIMA(p,d,q)(P,D,Q)^s$ 中的 p、q、P、Q 之前，必须先在差分运算过程中确定 d、D、s 的值。

最直观的方法是利用序列的散点图来了解时间序列的趋势性和季节性。更准确的方法是利用自相关函数来分析。

如果时间序列的自相关函数 r_k 在时滞 k 增大时，迅速趋于 0，可认为该序列是平稳的；如果随 k 的增大，r_k 不趋于 0，则序列是非平稳的；如果在 k=12（或 4），24（或 8），…，时，自相关函数显著不为 0，意味着时序具有季节性。

具有趋势性和（或）季节性的时间序列，可以经过差分运算处理转化为平稳序列，使其能使用 ARMA 模型来描述。经过 d 阶和（或）D 阶 s 步差分运算后，时间序列的自相关函数 r_k 在 k 增大时，迅速衰减并趋于 0，那么这里的 d 和（或）D、s 就是要识别的差分阶数和步数。

在实际应用中，d 和 D 的数值通常为 0、1 或 2，s 的数值通常为 4 或 12。

2. 确定模型

确定模型的过程，也就是确定 $ARIMA(p,d,q)(P,D,Q)^s$ 中的 p、q、P、Q。这是通过对时间序列的自相关函数及偏自相关函数的分析来实现的。

（1）确定 p 和 q

如果时间序列的自相关函数拖尾，而偏自相关函数在 p 步截尾，则序列可建立 AR 模型，其自回归阶数为 p。

如果序列的偏自相关函数拖尾，自相关函数在 q 步截尾，则序列可建立 MA 模型，其移动平均阶数为 q。

如果序列的自相关和偏自相关函数都拖尾，则要建立 ARMA 模型。这时，模型的阶数的识别比较复杂，通常要采用由低阶向高阶逐步试探的方法。几个可供参考的经验如下。

自回归阶数 p 的选择：具有统计有效性的偏自相关数目，或者有效偏自相关的时滞数。

移动平均阶数 q 的选择：显著不为 0 的自相关数目，或者自相关函数从 $k=k_0$ 开始迅速衰减，则 $q=k_0$。

（2）确定 P、Q、s

在分析序列的季节性时，首先分析其季节周期 s，可以通过分析序列散点图来确定 s 的值。确定 P 和 Q 的方法与确定 p 和 q 的方法相同，只是在观察自相关函数和偏自相关函数时，只分析 $k=s$，$2s$，…时的情形。

如果序列的自相关函数随时滞 k 的增大按照季节周期的增加而衰减，序列可以用季节自回归模型来描述，其阶数 P 决定于有效的季节偏自相关系数的数量。

如果序列的偏自相关函数随时滞 k 的增大按照季节周期的增加而衰减，序列可以用季节移动平均模型来描述，其阶数 Q 决定于显著不为 0 的季节自相关系数的数量。

如果序列的季节自相关函数和偏自相关函数都呈指数衰减，可以用季节的 ARMA 模型来描述。P 和 Q 的确定方法同上，但通常需要进行大量的测试，一般情况下，P 和 Q 的值为 0、1 或 2。

3．参数估计

确定模型之后，可以对选定的某一组阶数进行模型的参数估计。参数估计一般分两个步骤。第一步设法找出参数的初步估计；第二步在初步估计的基础上，根据一定的估计准则，例如最小二乘或者极大似然准则，求得模型参数在某种意义下的精确估计。这里仅介绍参数的初步估计。

（1）AR(p)模型参数的初步估计

AR(p)模型的公式为：

$$Y_t=\varphi_1 Y_{t-1}+\varphi_2 Y_{t-2}+\cdots+\varphi_p Y_{t-p}+e_t$$

对于 k=1，2，⋯，p，上式两边同乘 Y_{t-k}，可得：

$$Y_t Y_{t-k}=\varphi_1 Y_{t-1}Y_{t-k}+\varphi_2 Y_{t-2}Y_{t-k}+\cdots+\varphi_p Y_{t-p}Y_{t-k}+e_t Y_{t-k}$$

所以，

$$E(Y_t Y_{t-k})=\varphi_1 E(Y_{t-1}Y_{t-k})+\varphi_2 E(Y_{t-2}Y_{t-k})+\cdots+\varphi_p E(Y_{t-p}Y_{t-k})$$

即

$$r_k=\varphi_1 r_{k-1}+\varphi_2 r_{k-2}+\cdots+\varphi_p r_{k-p}$$

因为 k=1，2，⋯，p，所以上式展开得到方程组：

$$\begin{cases} r_1=\varphi_1+\varphi_2 r_1+\cdots+\varphi_p r_{p-1} \\ r_2=\varphi_1 r_1+\varphi_2+\cdots+\varphi_p r_{p-2} \\ \qquad\vdots \\ r_p=\varphi_1 r_{p-1}+\varphi_2 r_{p-2}+\cdots+\varphi_p \end{cases}$$

上式也称为 Yule-Walker 方程，根据此方程组即可求出参数 φ_1，φ_2，⋯，φ_p。

可以推出，对于 1 阶自回归模型 AR(1)，$\varphi_1=r_1$。

对于 2 阶自回归模型 AR(2)，有

$$\begin{cases} r_1=\varphi_1+\varphi_2 r_1 \\ r_2=\varphi_1 r_1+\varphi_2 \end{cases}$$

可解得：

$$\varphi_1=\frac{r_1(1-r_2)}{1-r_1^2}，\quad \varphi_2=\frac{r_2-r_1^2}{1-r_1^2}$$

（2）MA(q) 模型参数的初步估计

MA(q) 模型的公式为

$$Y_t=e_t-\theta_1 e_{t-1}-\theta_2 e_{t-2}-\cdots-\theta_q e_{t-q}$$

对于时滞 t–k，有 $Y_{t-k}=e_{t-k}-\theta_1 e_{t-k-1}-\theta_2 e_{t-k-2}-\cdots-\theta_q e_{t-k-q}$，所以有

$Y_t Y_{t-k}=(e_t-\theta_1 e_{t-1}-\theta_2 e_{t-2}-\cdots-\theta_q e_{t-q})(e_{t-k}-\theta_1 e_{t-k-1}-\theta_2 e_{t-k-2}-\cdots-\theta_q e_{t-k-q})$ 与 AR(p) 模型的初步估计的推导类似，上式两边同时取期望值，最终可得到

$$r_k=\frac{-\theta_k+\theta_1\theta_{k+1}+\theta_2\theta_{k+2}+\cdots+\theta_{q-k}\theta_q}{1+\theta_1^2+\theta_2^2+\cdots+\theta_q^2}$$

特别地，对于 MA(1)模型，有 $r_1=\dfrac{-\theta_1}{1+\theta_1^2}$，即 $r_1\theta_1^2+\theta_1+r_1=0$，解方程即可得到 MA(1)模型的估计参数：$\theta_1=\dfrac{-1\pm\sqrt{1-4r_1^2}}{2r_1}$。

对于 ARMA(p,q)模型参数的估计计算比较复杂，通常只能通过计算机软件来实现。

4．模型检验

模型的检验就是要考察所建立的模型是否合理。这个过程是通过对模型残差序列 e_t 检验是否白噪声，考核所建模型的优劣。残差序列就是观测时间序列与预测模型之间的误差序列。所谓白噪声序列，就是序列的各项之间互不相关的纯随机序列。如果残差序列 e_t 是白噪声序列，可认为模型合理，适用于预测；否则应进一步改进模型。模型的检验有自相关函数检验和 χ^2 检验两种方法。

（1）自相关函数检验法

一个纯随机序列的自相关系数在理论上应该是 0。但事实上，这只是在样本容量无限大时才成立。从总体中随机抽取一定数目的样本构成的序列，其自相关系数随样本的不同会有所差异，围绕着总体自相关系数构成一种分布。数理统计的理论证明，随机序列自相关系数的抽样分布，近似于以 0 为均值，$1/\sqrt{n}$ 为标准差的正态分布（n 为样本数量）。给定概率 $F(t)$，可以构成一个置信区间 $(-t\sigma,+t\sigma)$。例如 n=60，标准差 $\sigma=1/\sqrt{60}=0.129$，那么给定置信度 95%，可以构造出置信区间为（–1.96×0.129，+1.96×0.129）。如果残差序列的自相关系数基本都落入置信区间内，即可判定其为随机序列。

（2）χ^2 检验法

利用序列的自相关函数来分析序列的随机性，这种方法虽然简便直观，但检验精度有时并不理想。博克斯和皮尔斯提出了一种简单且精确度较高的模型检验法。

首先构造 Q 统计量，

$$Q=n\sum_{k=1}^{m}r_k^2$$

其中，m 为模型的最大时滞，n 为时间序列的观察值数量。

对于给定的置信概率 $1-\alpha$（通常取 α=0.05），查 χ^2 分布表中的自由度为 m 的 χ^2 值 $\chi_\alpha^2(m)$，将 Q 与 $\chi_\alpha^2(m)$ 进行比较，

如果 $Q\leqslant\chi_\alpha^2(m)$，则通过检验，残差序列为白噪声序列。

如果 $Q>\chi_\alpha^2(m)$，则未通过检验。

5．预测

在预测时人们总是追求预测的结果达到最优。所谓最优，就是预测值与实际值之间的误差尽可能小。

设 $\hat{Y}_t(l)$ 表示在已知 Y_t，Y_{t-1}，… 的条件下，对 Y_{t+l} 做出的预测，称为 l 步预测，其预测误差为：$e_t(l)=Y_{t+l}-\hat{Y}_t(l)$。

如果预测结果使得预测误差的方差达到最小，即 $E[e_t(l)]^2 = E[Y_{t+l} - \hat{Y}_t(l)]^2 = \min$，且预测值是过去时间序列值的函数，且是线性的，则称预测值 $\hat{Y}_t(l)$ 是 l 步线性最小方差预测，或者 l 步最优预测。

最优预测值可以用条件数学期望来表示，即：$\hat{Y}_t(l) = E(Y_{t+l} \mid Y_t, Y_{t-1}, \cdots, Y_1)$，它具有以下 3 个性质：

（1）$E\left(\sum_{j=1}^{l} \varphi_j Y_{t+j} \mid Y_t, Y_{t-1}, \cdots, Y_1\right) = \sum_{j=1}^{l} \varphi_j Y_{t+j} \mid Y_t, Y_{t-1}, \cdots, Y_1$

（2）$E(Y_{t+l} \mid Y_t, Y_{t-1}, \cdots, Y_1) = \begin{cases} \hat{Y}_t(l) & l > 0 \\ Y_{t+l} & l \leqslant 0 \end{cases}$

（3）$E(e_{t+l} \mid e_t, e_{t-1}, \cdots, e_1) = \begin{cases} 0 & l > 0 \\ e_{t+l} & l \leqslant 0 \end{cases}$

性质（1）说明条件期望是一种线性运算；性质（2）说明现在或者过去的观察值的条件期望就是它本身，未来观察值的条件期望就是它的预测值；性质（3）说明现在或过去的“误差”其条件期望就是它本身，未来误差的条件期望为 0。

根据上述性质，可以导出 ARMA(p,q)模型的预测公式。

一步预测公式为

$$\begin{aligned}\hat{Y}_t(1) &= E(Y_{t+1} \mid Y_t, Y_{t-1}, \cdots, Y_1) \\ &= E(\sum_{j=1}^{p} \varphi_j Y_{t-j+1} + e_{t+1} - \sum_{j=1}^{q} \theta_j e_{t-j+1} \mid Y_t, Y_{t-1}, \cdots, Y_1) \\ &= \hat{\varphi}_1 Y_t + \hat{\varphi}_2 Y_{t-1} + \cdots + \hat{\varphi}_p Y_{t-p+1} - \hat{\theta}_1 e_t - \hat{\theta}_2 e_{t-1} - \cdots - \hat{\theta}_q e_{t-q+1}\end{aligned}$$

其中，e_t，e_{t-1}，$\cdots$ 是可以计算的观测残差。

二步预测公式为

$$\begin{aligned}\hat{Y}_t(2) &= E(Y_{t+2} \mid Y_t, Y_{t-1}, \cdots, Y_1) \\ &= E(\sum_{j=1}^{p} \varphi_j Y_{t-j+2} + e_{t+2} - \sum_{j=1}^{q} \theta_j e_{t-j+2} \mid Y_t, Y_{t-1}, \cdots, Y_1) \\ &= \hat{\varphi}_1 \hat{Y}_t(1) + \hat{\varphi}_2 Y_t + \cdots + \hat{\varphi}_p Y_{t-p+2} - \hat{\theta}_2 e_t - \cdots - \hat{\theta}_q e_{t-q+2}\end{aligned}$$

类似地，可以求出 l 步预测公式

$$\hat{Y}_t(l) = \hat{\varphi}_1 \hat{Y}_t(l-1) + \hat{\varphi}_2 \hat{Y}_t(l-2) + \cdots + \hat{\varphi}_p Y_{t-p+l} - \hat{\theta}_l e_t - \cdots - \hat{\theta}_q e_{t-q+l}$$

特别地，当 $l>p$，$l>q$ 时，

$$\hat{Y}_t(l) = \hat{\varphi}_1 \hat{Y}_t(l-1) + \hat{\varphi}_2 \hat{Y}_t(l-2) + \cdots + \hat{\varphi}_p \hat{Y}_t(l-p)$$

9.3.4 在 Clementine 中应用时间序列分析

本小节描述一个根据数据样本集来建立时间序列模型并对序列的未来值进行预测的案例。在本例中，样本数据是我国 1999 年 12 月至 2009 年 9 月共 118 个月度的国家外汇

储备量的观察值（表 9-6），存放在“国家外汇储备规模.xls”文件中。我们要根据这批数据建立时间序列模型，并预测 2009 年 10 月至 2010 年 1 月共 4 个月的预测值。

表 9-6　国家外汇储备规模（亿美元）

月度	金额	月度	金额	月度	金额	月度	金额
99 年 12 月	1 546.75	02 年 6 月	2 427.63	04 年 12 月	6 099.32	07 年 6 月	13 326.25
00 年 1 月	1 561.00	02 年 7 月	2 465.34	05 年 1 月	6 236.46	07 年 7 月	13 852.00
00 年 2 月	1 565.59	02 年 8 月	2 530.90	05 年 2 月	6 426.10	07 年 8 月	14 086.41
00 年 3 月	1 568.20	02 年 9 月	2 586.30	05 年 3 月	6 591.44	07 年 9 月	14 336.11
00 年 4 月	1 568.46	02 年 10 月	2 655.39	05 年 4 月	6 707.74	07 年 10 月	14 548.98
00 年 5 月	1 580.19	02 年 11 月	2 746.25	05 年 5 月	6 910.12	07 年 11 月	14 969.06
00 年 6 月	1 585.68	02 年 12 月	2 864.07	05 年 6 月	7 109.73	07 年 12 月	15 282.49
00 年 7 月	1 585.96	03 年 1 月	3 044.60	05 年 7 月	7 327.33	08 年 1 月	15 898.10
00 年 8 月	1 592.17	03 年 2 月	3 082.50	05 年 8 月	7 532.09	08 年 2 月	16 471.34
00 年 9 月	1 600.92	03 年 3 月	3 160.10	05 年 9 月	7 690.04	08 年 3 月	16 821.77
00 年 10 月	1 613.44	03 年 4 月	3 262.91	05 年 10 月	7 849.02	08 年 4 月	17 566.55
00 年 11 月	1 639.11	03 年 5 月	3 400.61	05 年 11 月	7 942.23	08 年 5 月	17 969.61
00 年 12 月	1 655.74	03 年 6 月	3 464.76	05 年 12 月	8 188.72	08 年 6 月	18 088.28
01 年 1 月	1 686.23	03 年 7 月	3 564.86	06 年 1 月	8 451.80	08 年 7 月	18 451.64
01 年 2 月	1 747.73	03 年 8 月	3 647.34	06 年 2 月	8 536.72	08 年 8 月	18 841.53
01 年 3 月	1 758.47	03 年 9 月	3 838.63	06 年 3 月	8 750.70	08 年 9 月	19 055.85
01 年 4 月	1 771.78	03 年 10 月	4 009.92	06 年 4 月	8 950.40	08 年 10 月	18 796.88
01 年 5 月	1 790.00	03 年 11 月	4 203.61	06 年 5 月	9 250.20	08 年 11 月	18 847.17
01 年 6 月	1 808.38	03 年 12 月	4 032.51	06 年 6 月	9 411.15	08 年 12 月	19 460.30
01 年 7 月	1 844.92	04 年 1 月	4 157.20	06 年 7 月	9 545.50	09 年 1 月	19 134.56
01 年 8 月	1 900.53	04 年 2 月	4 266.39	06 年 8 月	9 720.39	09 年 2 月	19 120.66
01 年 9 月	1 957.64	04 年 3 月	4 398.22	06 年 9 月	9 879.28	09 年 3 月	19 537.41
01 年 10 月	2 030.29	04 年 4 月	4 490.17	06 年 10 月	10 096.26	09 年 4 月	20 088.80
01 年 11 月	2 083.15	04 年 5 月	4 585.60	06 年 11 月	10 387.51	09 年 5 月	20 894.91
01 年 12 月	2 121.65	04 年 6 月	4 706.39	06 年 12 月	10 663.44	09 年 6 月	21 316.06
02 年 1 月	2 174.00	04 年 7 月	4 829.82	07 年 1 月	11 046.92	09 年 7 月	21 746.18
02 年 2 月	2 235.31	04 年 8 月	4 961.69	07 年 2 月	11 573.72	09 年 8 月	22 108.27
02 年 3 月	2 276.05	04 年 9 月	5 145.38	07 年 3 月	12 020.31	09 年 9 月	22 725.95
02 年 4 月	2 338.24	04 年 10 月	5 424.43	07 年 4 月	12 465.66		
02 年 5 月	2 384.73	04 年 11 月	5 738.82	07 年 5 月	12 926.71		

完整的数据流如图 9.12 所示。

首先，将“数据源”中的 Excel 节点添加到数据流区域，并将“国家外汇储备规模.xls”文件加载到该节点，在该节点编辑窗口的“类型”标签下，将字段“月度”的方向设置为“无”，“金额”的方向设置为“输出”，如图 9.13 所示。然后单击“读取值”按钮，最后单击“确定”按钮。

向数据流中添加“时间区间”节点，并建立从“国家外汇储备规模.xls”节点到“时

间区间”节点的连接。在应用“时间序列”节点时，不能简单地将“时间序列”节点插入数据流并执行流。通常在“时间序列”节点之前，必须先插入“时间区间”节点，该节点可指定如下信息，例如所使用的时间区间（年、季度、月等）、用于估计的数据以及预测所延伸到的未来时间的范围（如果已使用）。或者说，只有在数据经过了“时间区间”节点的设置处理之后，Clementine 才将其视为时间序列数据来处理。

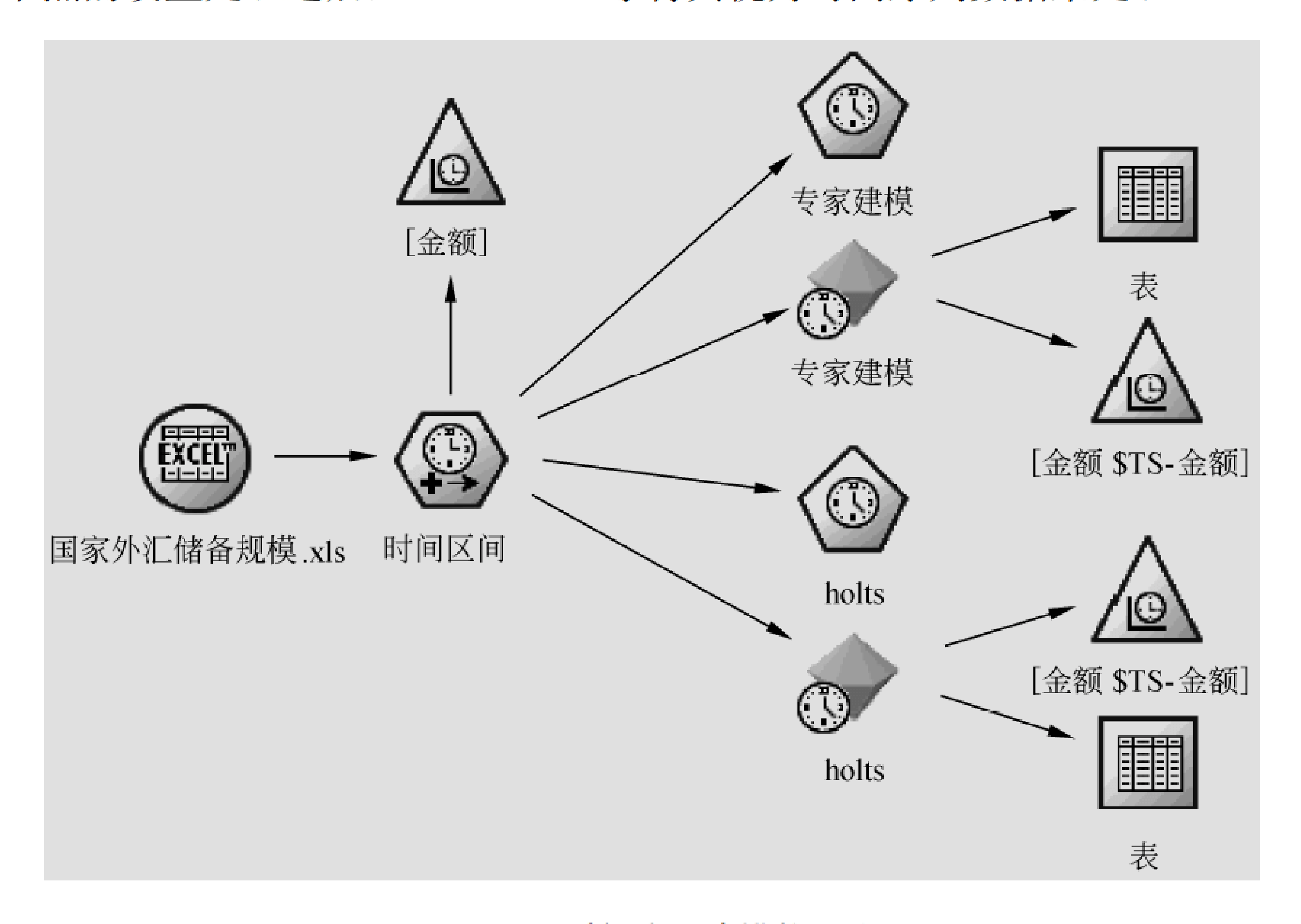


图 9.12　时间序列建模数据流



图 9.13　数据类型的设置

在“时间区间”节点的编辑窗口中，在“间隔”标签下，将“时间区间”设置为“月”，起始数据设置为“年：1999　月：十二月”，如图 9.14 所示。



图 9.14　设置时间区间参数

然后切换到“预报”标签下，勾选“将记录扩展至未来”复选框，并设置预测期数为“4”，即要预测未来 4 期的目标值，如图 9.15 所示。

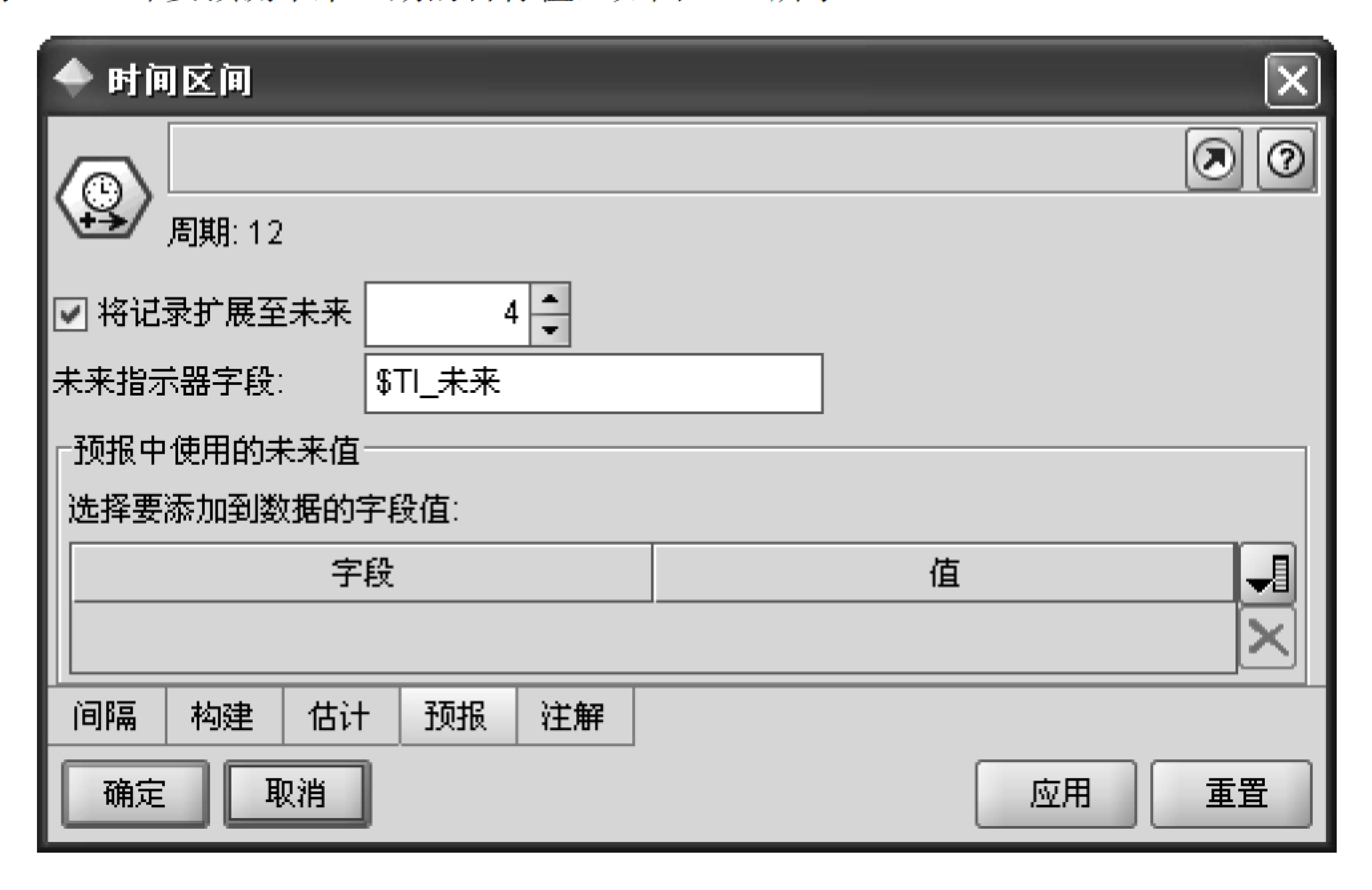


图 9.15　设置预报参数

在开始建模之前，先通过散点图来对这个时间序列特性进行初步的了解。将“时间散点图”节点添加到数据流中，并建立从“时间区间”到该“时间散点图”节点的连接。打开“时间散点图”节点的编辑窗口（如图 9.16 所示），将“序列”选定为“金额”，其他参数保持默认设置，然后单击“执行”按钮，即可得到如图 9.17 所示的时间散点图。



图 9.16 设置时间散点图

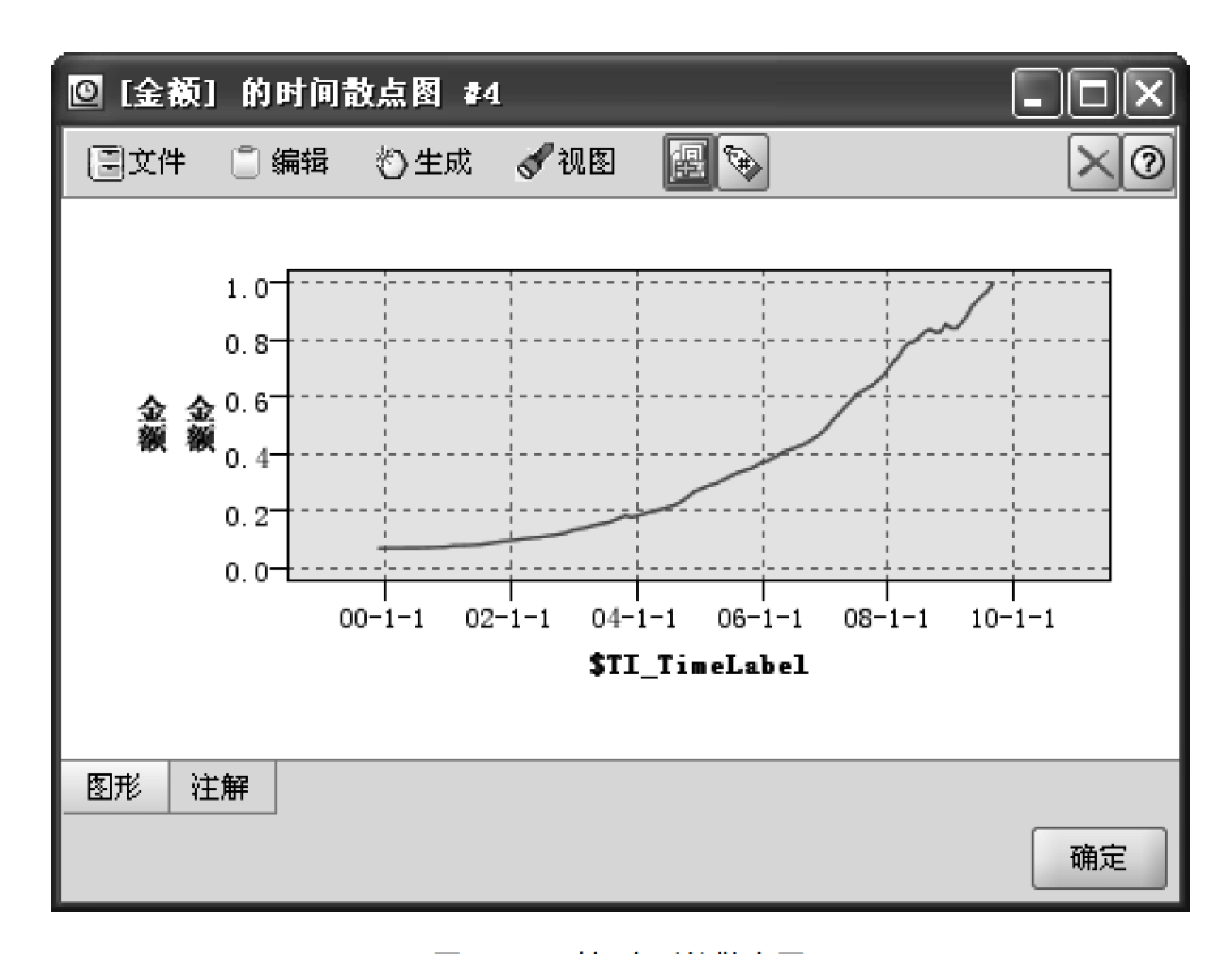


图 9.17 时间序列的散点图

从散点图可以看出，该时间序列具有明显的指数增长上升趋势，但不包含季节性。

将“时间序列”节点添加到数据流中，并建立从“时间区间”节点到该“时间序列”节点的连接。打开该“时间序列”节点的编辑窗口，做如图 9.18 所示的设置。

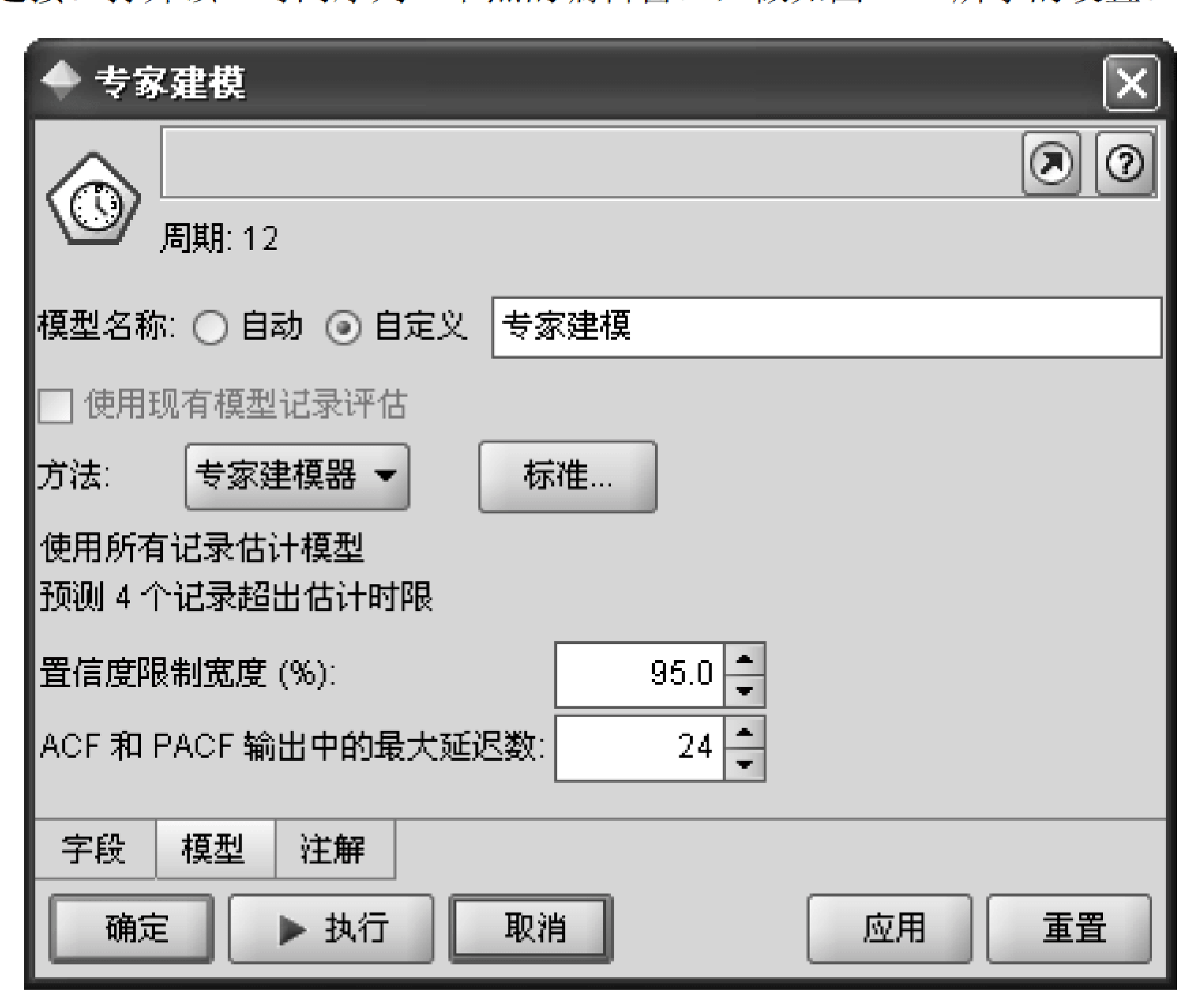


图 9.18　设置“专家建模”节点

将模型名称自定义为“专家建模”，把建模的“方法”设置为“专家建模器”。Clementine 为我们提供了 3 种建模的方式。

专家建模器（Expert Modeler）：选择此选项将自动为时间序列查找拟合得最好的模型。

指数平滑（Exponential Smoothing）：使用此选项以指定自定义的指数平滑模型。

ARIMA：使用此选项以指定自定义的 ARIMA 模型。

单击“标准...”按钮，设置模型的类型，如图 9.19 所示。

所有模型（All Models）：如果选择该选项，专家建模器将同时考虑 ARIMA 模型和指数平滑模型。

仅限于指数平滑的模型（Exponential Smoothing Models Only）：如果选择该选项，专家建模器仅考虑指数平滑模型。

仅限于 ARIMA 模型（ARIMA Models Only）：如果选择该选项，专家建模器仅考虑 ARIMA 模型。

专家建模器考虑季节模型（Expert Modeler Considers Seasonal Models）：仅在已为活动数据集定义了周期性时才启用此选项。选中此选项时，专家建模器将同时考虑季节模型和非季节模型。如果未选中此选项，则仅考虑非季节模型。

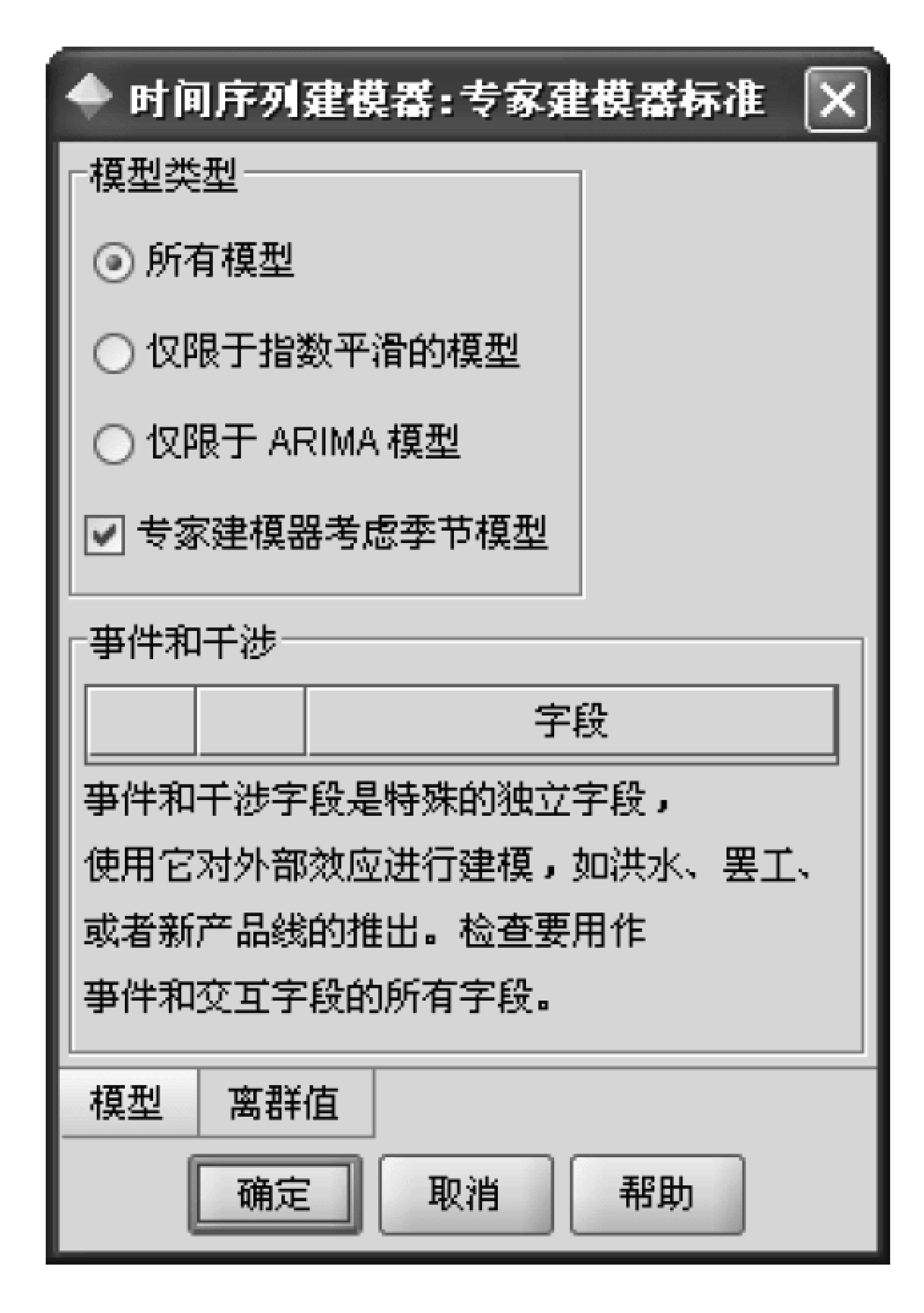


图 9.19　设置时间序列模型类型

最后，设置“置信度限制宽度”和“ACF 和 PACF 输出中的最大延迟数”（保持默认设置）。

置信度限制宽度（Confidence Limit Width）：计算模型预测和残差自相关的置信区间。可以指定任何小于 100 的正数。默认情况下，将使用 95%置信区间。

ACF 和 PACF 输出中的最大延迟数（Maximum Number of Lags In ACF and PACF Output）：可设置在自相关和偏自相关的表和散点图中显示的最大延迟数，即最大时滞。

以上设置全部结束后，单击“执行”按钮，即可在管理器窗口的“模型”标签下显示生成的“专家建模”模型。将生成的“专家建模”模型拖入到数据流区域，并建立从“时间区间”节点到“专家建模”的连接。

双击生成的“专家建模”节点，可以浏览该模型的详细信息。在“模型”标签下显示了模型的基本信息，如图 9.20 所示。

从图 9.20 中可以看到，专家建模器为我们建立了一个 ARIMA(1,1,0)(0,0,0)模型，也就是一个经过 1 阶差分的 AR(1)模型。

预测变量为 0。时间序列节点可为时间序列建立单变量 ARIMA 模型，也可以在模型中包括预测变量来建立多变量 ARIMA 模型。本例建立的是单变量 ARIMA 模型，所以预测变量为 0。

固定的 R**2，即 stationary R-squared，是模型的拟合优度测度。此统计量是序列中由模型解释的总变异所占比例的估计值。该值越高（最大值为 1.0），则模型拟合会越好。此时单击窗口中“视图”下拉列表框，选择“高级”选项，可以显示其他拟合优度测度值，如 RMSE（均方根误差）、MAPE（平均绝对百分误差）、MAE（平均绝对误差）、

MaxAPE（最大绝对百分比误差）、MaxAE（最大绝对误差）等。如前文中所述，预测准确度通常以预测误差的方差最小为准则，所以可以重点参考 $\mathrm{MAE}=\frac{1}{n}\sum|Y(t)-\hat{Y}(t)|$ 的值（这里 n 为残差的个数）。本例中，“专家建模”模型的 MAE=80.143。



图 9.20　专家建模基本信息

Q=17.311，是对残差序列的随机性进行 χ^2 检验的 Q 统计量，df=17 是自由度；Sig.=0.433 是 Q 统计量的显著性值（如果显著性值小于 0.05，表示残差序列不是随机的，则意味着所观测的序列中存在模型无法解释的结构，也就是说所建立的模型并不理想）。

切换到“参数”标签，如图 9.21 所示。



图 9.21　模型的参数

在“评估”列中显示了模型中参数的估计值；SE 列显示了参数估计值的标准误差；t 列显示了参数估计值除以标准误差后的值；Sig.列显示了参数估计值的显著性级别，0.05 以上的值视为没有显著的统计意义。

切换到“残差”标签下，显示了残差序列的自相关分析图，如图 9.22 所示。

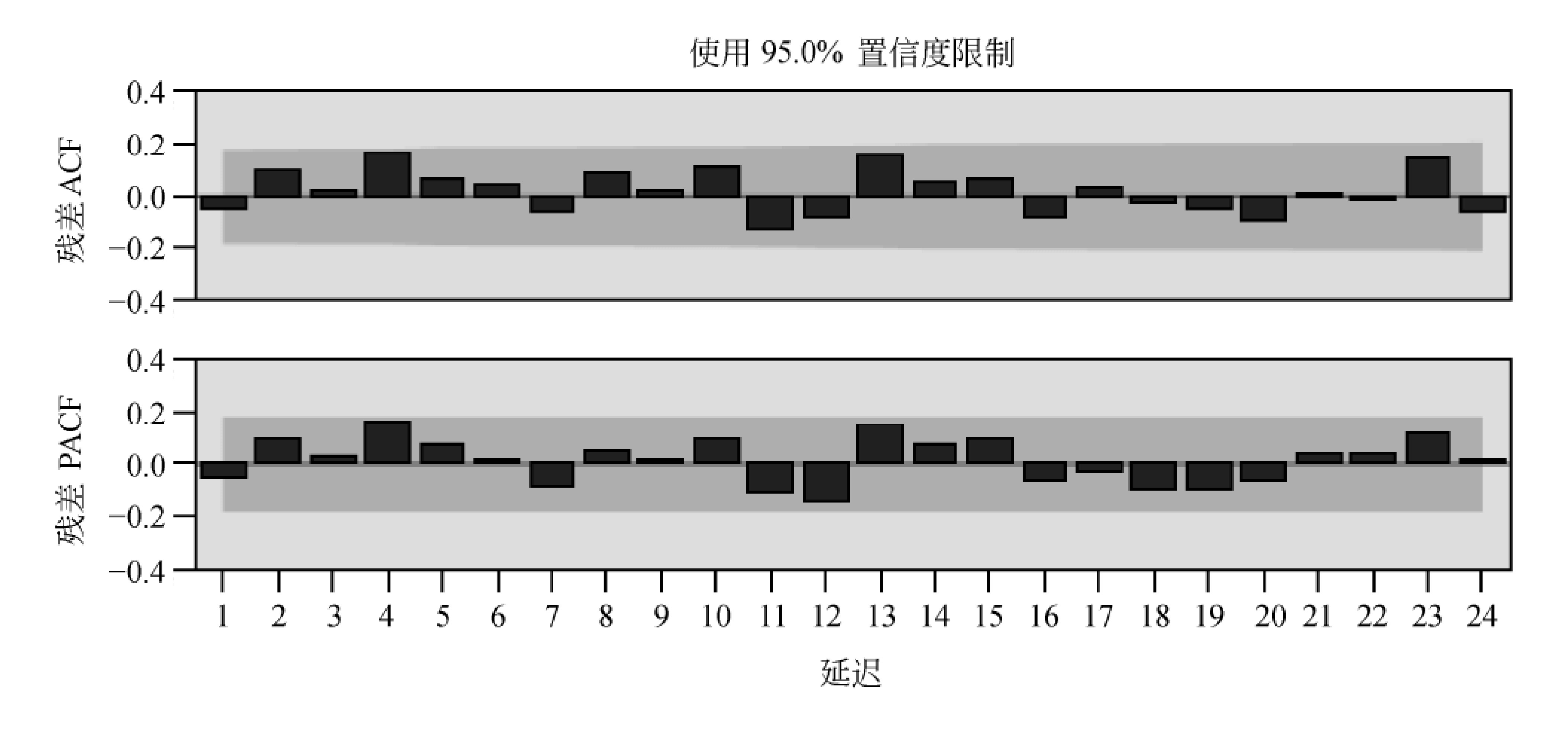


图 9.22　残差序列的自相关分析

从图中可以看出，残差序列的自相关系数全部落入 95%置信区间内，说明残差序列是随机序列。

下面来考察一下该模型对样本数据的拟合情况。将“时间散点图”节点添加到数据流中，并建立从生成的“专家建模”节点到该“时间散点图”节点的连接。对该“时间散点图”节点进行如图 9.23 所示的设置，然后单击“执行”按钮，即可得到如图 9.24 所示的时间散点图。



图 9.23　设置时间散点图节点

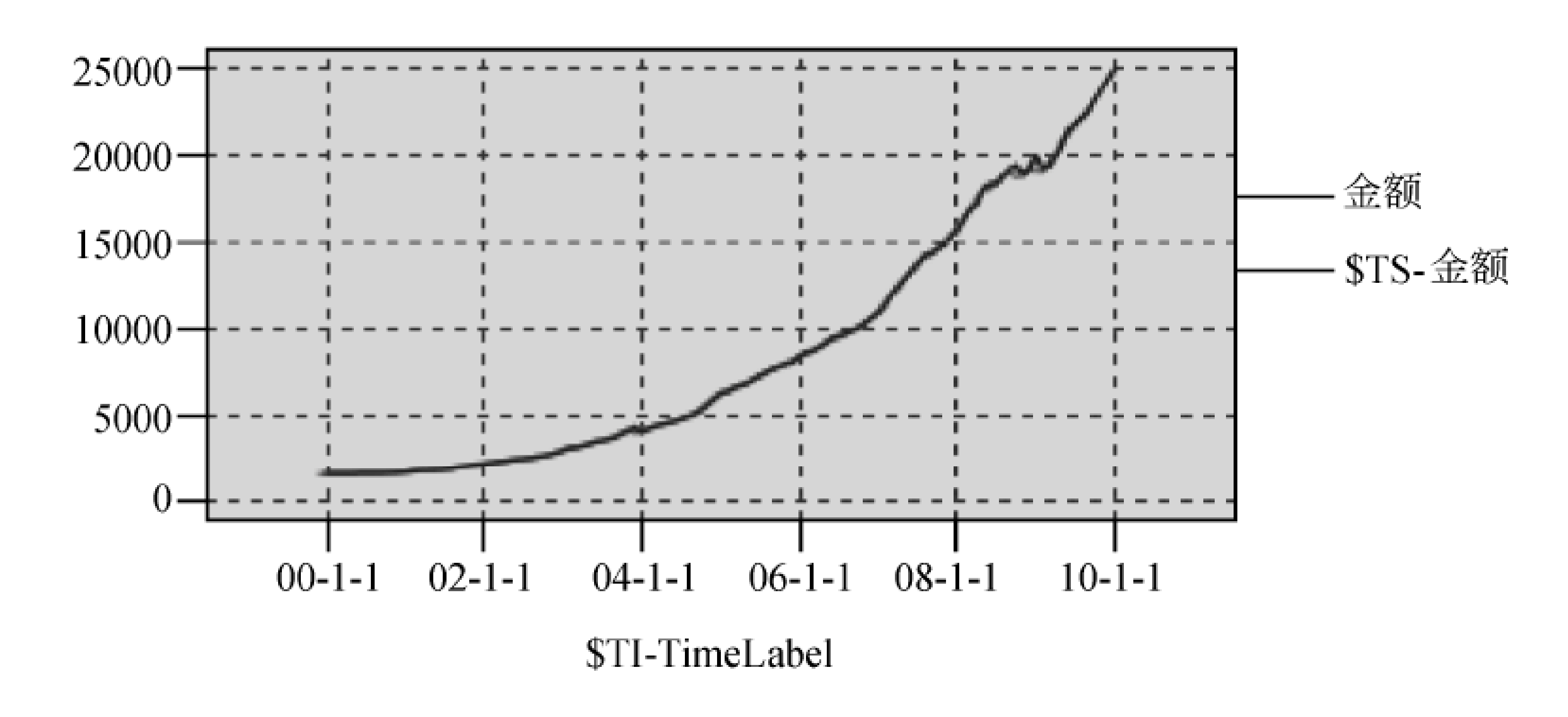


图 9.24 预测值与期望值的时间散点图

从图 9.24 中可以看到，预测值曲线基本上覆盖在期望值之上，说明模型对数据的拟合效果很好。

最后，向数据流区域中添加“表”节点，建立从生成的“专家建模”节点到“表”节点的连接，执行“表”节点，即可将预测结果输出到表中，如图 9.25 所示。

表（10 个字段，122 条记录）

文件 编辑 生成

	el...	$TI_TimeL...	$TI_Year	$TI_...	$TI_...	$TS-金额	$TSLCI-金额	$TSUCI-金额
115	115	六月 2009	2009	6	0	21510.080	20920.136	22112.332
116	116	七月 2009	2009	7	0	21788.861	21191.271	22398.919
117	117	八月 2009	2009	8	0	22228.697	21619.044	22851.069
118	118	九月 2009	2009	9	0	22570.263	21951.242	23202.199
119	119	十月 2009	2009	10	1	23294.571	22536.065	24073.895
120	120	十一月 2009	2009	11	1	23853.812	22551.752	25216.911
121	121	十二月 2009	2009	12	1	24417.818	22651.222	26295.887
122	122	一月 2010	2010	1	1	24991.945	22815.020	27336.416

表 注解

确定

图 9.25 “专家建模”节点的输出结果

在输出结果中，“$TS-金额”是预测结果，表中的最后 4 行是对未来 4 期的预测值，即：

2009 年 10 月：23 294.571　　2009 年 11 月：23 853.812

2009 年 12 月：24 417.818　　2010 年 1 月：24 991.945

下面再利用指数平滑方法建立另一个时间序列模型。将“时间序列”节点添加到数据流中，并建立从“时间区间”节点到该“时间序列”节点的连接。打开该“时间序列”节点的编辑窗口，做如图 9.26 所示的设置。

将建模方法设置为“指数平滑”，然后单击“标准...”按钮，将模型类型设置为“Holts 线性趋势”，“目标转换”设置为“自然对数”（将指数增长趋势转换为线性增长趋势）。

设置结束后即可执行，得到生成的 holts 模型。

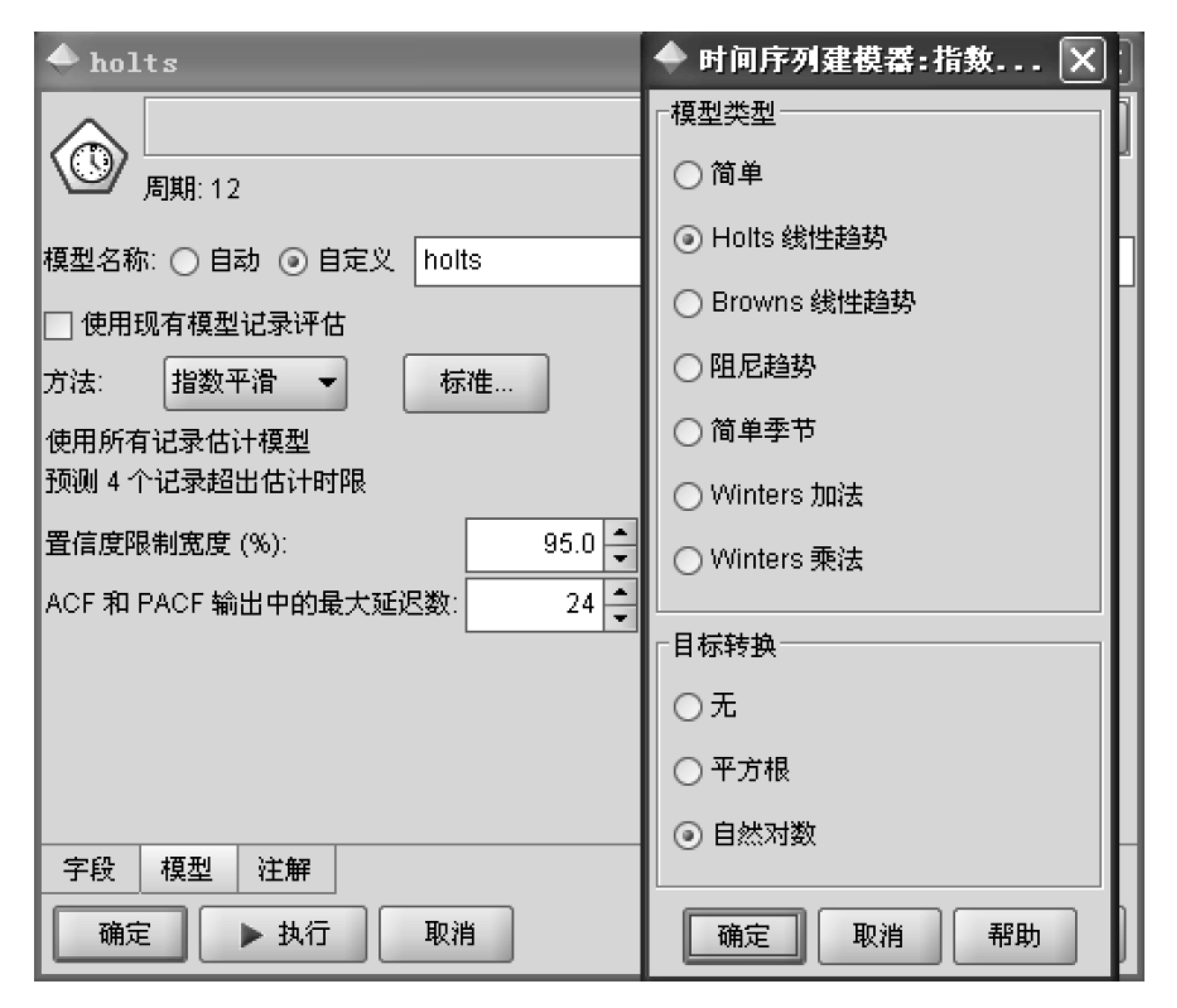


图 9.26　设置指数平滑建模参数

模型的浏览与预测与前面的描述基本一致，这里不再赘述。预测结果显示，holts 模型的 MAE=85.052，大于前面“专家建模”模型的 MAE=80.143。所以，从 MAE 的比较来看，“专家建模”模型更优。

参 考 文 献

[1] Krzysztof Cios, Lukasz Kurgan. Trends in Data Mining and Knowledge Discovery. In: Pal N.R.,Jain, L.C. and Teoderesku, N. (Eds.), Knowledge Discovery in Advanced Information Systems,Springer. 2002

[2] 李强. 创建决策树算法的比较研究——ID3，C4.5，C5.0 算法的比较.甘肃科学学报，2006，18(4)

[3] Data Mining Tools See5 and C5.0. http://www.rulequest.com/see5-info.html

[4] P.Smyth, R.M.Goodman. An Information Theoretic Approach to Rule Induction from Databses. IEEE Transactions on knowledge and data engineering, 1992, 4(4)

[5] Saltelli, A., S. Tarantola, F. , F. Campolongo, and M. Ratto. Sensitivity Analysis in Practice—A Guide to Assessing Scientific Models. : JohnWiley. 2004

[6] Saltelli, A. 2002. Making best use of model evaluations to compute sensitivity indices. Computer Physics Communications, 145(2)：280～297

[7] Leo Breiman, Jerome Friedman, R.A.Olshen, Charles J.Stone . Classification and Regression Trees. http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf

[8] Zhang Tian, Ramakrishnan R. BIRCH:A New Data Clustering Algorithm and Its Applications. Data Mining and Knowledge Discovery, 1997(1)

[9] Tian Zhang, Raghu Ramakrishnan, Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. SIGMOD '96 6/96 Montreal, Canada

[10] 梁循. 数据挖掘算法与应用. 北京：北京大学出版社，2006

[11] 章兢，张小刚. 数据挖掘算法及其工程应用. 北京：机械工业出版社，2006

[12] Agrawal R., Imielienski T., Swami A. Mining Association Rules between Sets of Items in Large Databases. In: Proc. Conf. on Management of Data, 207～216. New York: ACM Press. 1993

[13] Agrawal A.,Mannila H., Srikant R., Toivonen H., and Verkamo A. Fast Discovery of Association Rules. In: Fayyad et al. 1996：307～328

[14] C. Borgelt, R. Kruse, Induction of association rules: Apriori implementation. Presented at 15th Conference on Computational Statistics (Germany, 2002). http://fuzzy.cs.uni-magdeburg.de/~borgelt/papers/cstat_02.pdf

[15] Hidber C. Online association rule mining. Proceedings of ACM SIGMOD International Conference on Management of Data. 1999: 145～156

[16] 李琦，宋国新. 在线挖掘关联规则算法的改进. 华东理工大学学报，2002，26(5)

[17] Tsur D, Ullman J , Abiteboul S, et al. Query Flocks: A generalization of association-rule mining. Proceedings of ACM SIGMOD International Conference on Management of Data, 1998

[18] Agrawal R, Srikant R. Mining Sequential Patterns. Proc 1995 Int Conf Data Engineering (ICDE'95). Taipei : IEEEComputer Society ,1995: 3～141

[19] Agrawal R., Srikant R. Mining Sequential Patterns : Generalizationgs and Performance

Improvments. Proc 5th Int ConfExtending Database Technology (EDBT). Avignon : Lecture Notes in Computer Science, 1996: 3～17

[20] TALB IH,DRAA A,BATOUCHEM. A new quantum inspired genetic algorithm for solving the traveling salesman problem . Proc of IEEE International Conference on Industrial Technology, 2004: 1192～1197

[21] 范克新. 社会学定量方法. 南京：南京大学出版社，2004

[22] 贾俊平，金勇进. 统计学. 北京：中国人民大学出版社，2004

[23] 卢淑华. 社会统计学. 北京：北京大学出版社，2001

[24] 郝拉娣，于化东. 标准差与标准误. 编辑学报，2005，17(2)

[25] 王海燕，杨方廷，刘鲁. 标准化系数与偏相关系数的比较与应用. 数量经济技术经济研究. 2006，9

[26] 中国统计年鉴. 国家统计局 http://www.stats.gov.cn/ndsj/information/njml.html

[27] 蒋宗礼. 人工神经网络导论. 北京：高等教育出版社，2001

[28] 张立明. 人工神经网络的模型及其应用. 上海：复旦大学出版社，1993

[29] 陈祥光，裴旭东. 人工神经网络技术及其应用. 北京：中国电力出版社，2003

[30] 杨建刚. 人工神经网络实用教程. 杭州：浙江大学出版社，2001

[31] 高隽. 人工神经网络原理及仿真实例. 北京：机械工业出版社，2007

[32] 刘思峰，党耀国. 预测方法与技术. 北京：高等教育出版社，2005

[33] 吴清烈，蒋尚华. 预测与决策分析. 南京：东南大学出版社，2004.2

[34] 齐小华. 预测决策方法——在广告中的应用. 北京：北京广播学院出版社，2003

[35] 潘红宇. 时间序列分析. 北京：对外经济贸易大学出版社，2006

[36] 王燕. 应用时间序列分析. 北京：中国人民大学出版社，2005.7

[37] 徐国祥，胡清友. 统计预测和决策. 上海：上海财经大学出版社，2004

[38] 薛薇. SPSS 统计分析方法及应用. 北京：电子工业出版社，2009

[39] 易丹辉. 统计预测——方法与应用. 北京：中国统计出版社，2001

图 书 资 源 支 持

感谢您一直以来对清华版图书的支持和爱护。为了配合本书的使用，本书提供配套的素材，有需求的用户请到清华大学出版社主页(http://www.tup.com.cn)上查询和下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本书的过程中遇到了什么问题，或者有相关图书出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。

我们的联系方式：

地　　址：北京海淀区双清路学研大厦 A 座 707

邮　　编：100084

电　　话：010－62770175－4604

资源下载：http://www.tup.com.cn

电子邮件：weijj@tup.tsinghua.edu.cn

QQ：883604(请写明您的单位和姓名)

扫一扫

资源下载、样书申请

新书推荐、技术交流

用微信扫一扫右边的二维码，即可关注清华大学出版社公众号“书圈”。